

# Circuitos Digitais

Engenharia Elétrica/Engenharia de Automação/  
Engenharia de Computação/Sistemas de Informação/  
Ciência da Computação/Tecnologia de Redes

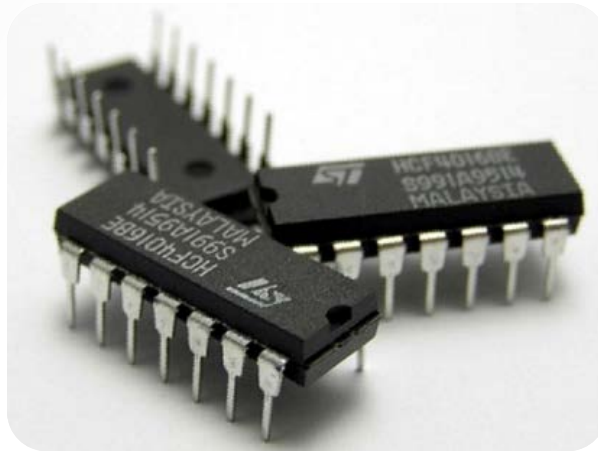


Prof. VICTOR MARQUES MIRANDA

# CONTEÚDOS

- I. Conceitos Básicos de Sistemas Digitais
- II. Sistemas de Numeração
  - II.1. Conversões entre Bases*
  - II.2. Operações Aritméticas*
- III. Portas Lógicas e Formas de Representação de uma Função Lógica
- IV. Álgebra Booleana e Simplificação de Circuitos
- V. Redes Combinacionais e Minimização Lógica
- VI. Projeto Lógico Combinacional
- VII. Módulos-Padrão Combinacionais e Aritméticos
- VIII. Sistemas Sequenciais – Parte 1: Máquinas de Estados, Elementos de Memória e Análise e Projeto de Redes Sequenciais Canônicas
- IX. Sistemas Sequenciais – Parte 2: Módulos-Padrão – Contadores
- X. Revisão dos Conteúdos e Aplicação da N2





## Unidade 2

# Sistemas de Numeração



# Objetivos

- ✓ Sistemas de Numeração
  - ✓ Bases Numéricas
  - ✓ Conversões entre Bases
  - ✓ Número Binário: Bits, Bytes e Palavras
  - ✓ Operações Aritméticas com Números Binários

# Introdução

# Introdução

- Como surgiram os números?
- Registros nos mostram que a primeira atividade de contagem está relacionado a atividade de pastoreio.
- Ao soltar os animais o pastor separava uma pedra para cada animal e com isso fazia um monte de pedras. Quando os animais voltavam o pastor retirava uma pedra para cada animal. Assim, sabia se estava faltando algum animal.
- Uma evidência prática da origem da palavra Cálculo do latim *Calculus*, que significa pedra.

# Introdução

- Depois dessa primeira noção de quantidade, surgiu a numeração escrita, do desejo de manter os registros que antes eram simbolizados pelas pedras.
- Essa numeração escrita era feita com marcas em madeiras ou qualquer outro objeto que possibilitasse a marcação. Vale destacar, que essas marcações são tão ou mais antigas que a própria escrita.
- Os primeiros sistemas de escrita numérica que se conhece são os dos egípcios e os dos sumérios, surgidos por volta de 3.500 a.C.

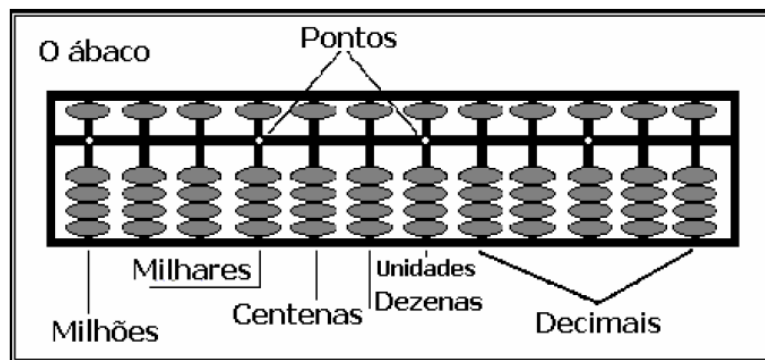
# Introdução

- Os sistemas são semelhantes: ambos atribuem símbolos aos números 1, 10, 100, 1000 etc. e fazem a representação dos outros como sendo a soma desses "principais". Então o número 354 era a soma de três cens, cinco dezenas e quatro uns.
- Depois dos símbolos, veio a ideia de representar os números com letras. Usado pelos povos hebraico e grego, tal sistema deu origem ao sistema romano onde os números 1, 5, 10, 50, 100, 500, 1000 são representados pelas letras I, V, X, L, C, D, M, respectivamente.



# Introdução

- Observe que a base tanto dos gregos como dos hebraicos era 10 devido aos dez dedos humanos. Nossa primeira máquina de fazer cálculo.
- Os romanos acrescentaram a 5 também (apenas uma mão!).
- Mas o cálculo ficou complexo e precisou-se evoluir.
- Assim surgiu a primeira máquina de calcular: Ábaco.



# Introdução

- Um ábaco adaptado, inventado por Helen Keller e chamado de *Cranmer*, é ainda utilizado por deficientes visuais.
- Foi mostrado que alunos chineses conseguem fazer contas complexas com um ábaco, mais rapidamente do que um ocidental equipado com uma moderna calculadora electrónica. Embora a calculadora apresente a resposta quase instantaneamente, os alunos conseguem terminar o cálculo antes mesmo de seu competidor acabar de digitar os algarismos no teclado da calculadora.

# Numeração Posicional

- Por volta do século V d.C., surge, na Índia, a **numeração posicional de base 10**, que usamos atualmente. Este sistema foi divulgado na Europa em torno de 825 d.C. pelo matemático árabe Mohamed Ben Mussa Al Khawarismi, por isso que o sistema ficou conhecido como sistema indo-arábico, pois surgiu na Índia.
- Na **numeração posicional**, cada dígito (algarismo) possui um peso de acordo com sua posição relativa ao LSD. Assim, o valor de um dado número corresponde a uma soma ponderada de seus dígitos. Os pesos desta ponderação são potências da base de numeração que se utiliza.

*Exemplo: **Sistema decimal:** O símbolo 5 expressa cinco objetos em 15 ou 1235, mas expressa cinquenta objetos em 458.*

- Já a **numeração não-posicional**, o valor de um símbolo é sempre o mesmo e independe da posição em que ocupa no número.

*Exemplo: **Numeração Romana:** O símbolo V expressa sempre cinco objetos, seja em IV, XV ou LXVII.*

# Numeração Posicional

- **Decimal:** dez símbolos diferentes (0 até 9).

**Exemplo: 2345**

Milhar	Centena	Dezena	Unidade
2	3	4	5
$2 \times 10^3$	$3 \times 10^2$	$4 \times 10^1$	$5 \times 10^0$
2.000	300	40	5

# Numeração Posicional

- A numeração posicional significou uma grande evolução no processo de cálculo, pois era a representação do mecanismo do ábaco.
- Observe a comparação da operação de somas dos mesmos números no sistema posicional de base dez e no sistema romano:

1432

MCDXXXII

2468

MMCDLXVIII

3900

MMMCM

Pense em multiplicação no sistema romano!?!?!?

# Bases Numéricas

# Bases Numéricas

- A base de um sistema de numeração posicional define o número de algarismos distintos utilizados para representar números.
- ✓ Um **sistema numérico de base B utiliza B símbolos distintos** para representar qualquer grandeza.
- ✓ **O menor valor representável** com um único símbolo em um sistema numérico de base B **é zero**.
- ✓ **O maior valor representável** com um único símbolo em um sistema numérico de base B **é  $B - 1$** .

# Bases Numéricas

## Principais Bases Numéricas para Computação:

- **Decimal:** dez símbolos diferentes (0 até 9).
- **Binário:** dois símbolos diferentes (0 e 1).
- **Hexadecimal:** dezesseis símbolos diferentes (0 até 9 e A até F).

**Obs:** Analisaremos também a base **Octal:** oito símbolos diferentes (0 até 7).



# Resumo de Conversões de Bases

- Conversão Binário → Decimal
- Conversão Decimal → Binário
- Conversão Hexadecimal → Decimal
- Conversão Decimal → Hexadecimal
- Conversão Hexadecimal → Binário
- Conversão Binário → Hexadecimal
- Conversão Octal → Binário
- Conversão Binário → Octal
- Conversão Octal → Decimal
- Conversão Decimal → Octal
- Conversão Octal → Hexadecimal
- Conversão Hexadecimal → Octal

# Base Decimal

- A base numérica mais usada atualmente no nosso dia-a-dia é a base decimal, ou seja, a base 10.
- Portanto, quando escrevemos **123**, por exemplo, estamos nos referindo ao número que contém três unidades, duas dezenas e uma centena, ou seja, de acordo com a numeração posicional, o número 123 nada mais é que uma abreviação da expressão “**123** =  $1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 = 100 + 20 + 3$ ”

# Conversão para a Base Decimal

- **Genericamente**, podemos dizer que um número da forma “ $a_n a_{n-1} a_{n-2} \dots a_2 a_1 a_0$ ”, no sistema de numeração decimal, pode ser escrito na forma:

$$D = a_n 10^n + a_{n-1} 10^{n-1} + a_{n-2} 10^{n-2} + \dots + a_2 10^2 + a_1 10 + a_0$$

- Mas será que o número precisa necessariamente estar na base 10?
- Tomamos um **número inteiro positivo** “ $x$ ” qualquer. “ $x$ ” pode ser escrito numa **base numérica** “ $r$ ” qualquer, e seu correspondente valor na **base decimal** pode ser escrito na forma:

$$(x)_{10} = x_n r^n + x_{n-1} r^{n-1} + x_{n-2} r^{n-2} + \dots + x_2 r^2 + x_1 r + x_0$$

# Conversão para a Base Decimal

Representação de Números por meio de vetor de dígitos em uma base qualquer  $r$ :

$x = (x_{n-1}, x_{n-2}, \dots, x_0)_r$ 
 onde  $x$  assume valores do conjunto  $(0, 1, \dots, r-1)$

MSD      LSD

$$x = \sum_{i=0}^{n-1} x_i r^i$$

As bases mais frequentemente usadas são as da forma:  $r = 2^k$

# Conversão para a Base Decimal

Exemplo:

$$x = (1, 0, 0, 1, 0, 1)_2$$

$$x = \sum_{i=0}^{n-1} x_i r^i$$

$$x = 1x2^5 + 0x2^4 + 0x2^3 + 1x2^2 + 0x2^1 + 1x2^0$$

$$X = (37)_{10}$$

# Conversão para a Base Decimal

Exemplo:

$$x = (1, 2, 1, 0)_4$$

$$x = \sum_{i=0}^{n-1} x_i r^i$$

$$X = (?)_{10}$$

# Conversão para a Base Decimal

Exemplo:

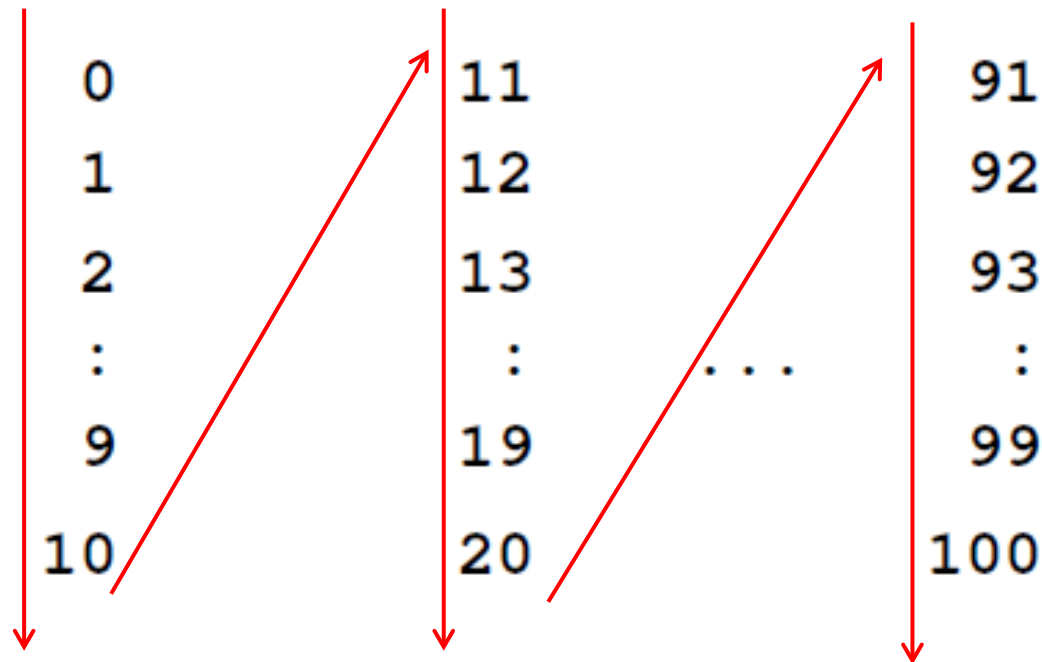
$$x = (1, 2, 1, 0)_4$$

$$x = \sum_{i=0}^{n-1} x_i r^i$$

$$x = 1 \times 4^3 + 2 \times 4^2 + 1 \times 4^1 + 0 \times 4^0$$

$$x = (100)_{10}$$

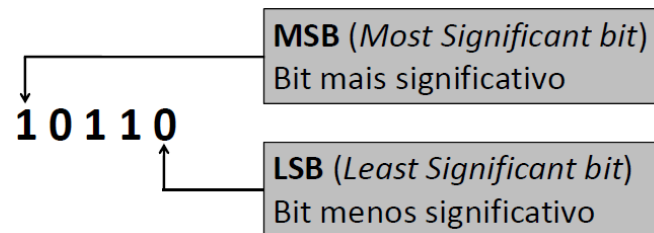
# Contagem em Decimal





# Base Binária

- Infelizmente, o **sistema decimal não é adequado aos sistemas digitais**, porque é muito difícil implementar circuitos eletrônicos que trabalhem com 10 níveis diferentes de tensão (cada nível representando um dígito decimal, de 0 a 9).
- Por outro lado, é muito fácil implementar circuitos eletrônicos que operem com dois níveis de tensão. Por isso, **quase todos os sistemas digitais usam o sistema de numeração binário (base 2)** como sistema básico para suas operações, embora outros sistemas também possam ser utilizados.
- No sistema binário existem somente dois símbolos ou dígitos, o “0” e o “1”, chamados **bit** (**binary digit**).
- Número Binário é uma sequência de bits.



# Base Binária

- Embora os computadores tenham instruções (ou comandos) que possam testar e manipular bits, geralmente são idealizados para armazenar instruções em múltiplos de bits, chamados **bytes**.
- Existem também termos para referir-se a múltiplos de bits usando padrões prefixados, como quilobit (**kb**), megabit (**Mb**), gigabit (**Gb**) e Terabit (**Tb**).
- A notação para bit utiliza um "*b*" minúsculo, em oposição à notação para **byte** que utiliza um "*B*" maiúsculo (**kB, MB, GB, TB**).

# Base Binária

- 1 Bit = Binary Digit
- 8 Bits = 1 Byte
- 16 bits = 2 Bytes

Múltiplos do byte					
Prefixo binário (IEC)			Prefixo do SI		
Nome	Símbolo	Múltiplo	Nome	Símbolo	Múltiplo
byte	B	$2^0$	byte	B	$10^0$
kibibyte	KiB	$2^{10}$	Kilobyte	kB	$10^3$
mebibyte	MiB	$2^{20}$	megabyte	MB	$10^6$
gibibyte	GiB	$2^{30}$	gigabyte	GB	$10^9$
tebibyte	TiB	$2^{40}$	terabyte	TB	$10^{12}$
pebibyte	PiB	$2^{50}$	petabyte	PB	$10^{15}$
exbibyte	EiB	$2^{60}$	exabyte	EB	$10^{18}$
zebibyte	ZiB	$2^{70}$	zettabyte	ZB	$10^{21}$
yobibyte	YiB	$2^{80}$	yottabyte	YB	$10^{24}$

# Conversão Binário ↔ Decimal

- Ex: 4 bits

Decimal	Binário
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

# Conversão Binário → Decimal

- **Conversão Inteira:**
  - Soma dos produtos de cada bit por seu peso relativo.

**Exemplo:  $1011_2$**

1	0	1	1
$1 \times 2^3$	$0 \times 2^2$	$1 \times 2^1$	$1 \times 2^0$
8	0	2	1
$8 + 0 + 2 + 1 = 11$			

# Conversão Binário → Decimal

- **Conversão de Binário para Decimal**

- Soma dos produtos de cada bit por seu peso relativo.

- **Exemplo 1:**

**Binário:** (1 1 0 1 1)<sub>2</sub>

**Decimal:** (1x2<sup>4</sup> + 1x2<sup>3</sup> + 0x2<sup>2</sup> + 1x2<sup>1</sup> + 1x2<sup>0</sup>)<sub>10</sub>

**Decimal:** (16 + 8 + 0 + 2 + 1)<sub>10</sub>

**Decimal:** (27)<sub>10</sub>

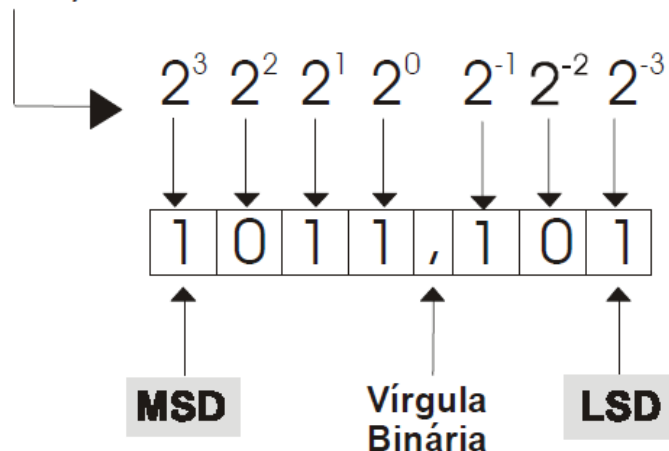
**Exemplo 2: Binário:** (010110101)<sub>2</sub>

**Decimal:** ?

# Conversão Binário → Decimal

## – Conversão Fracionária:

Valores Posicionais  
(pesos)

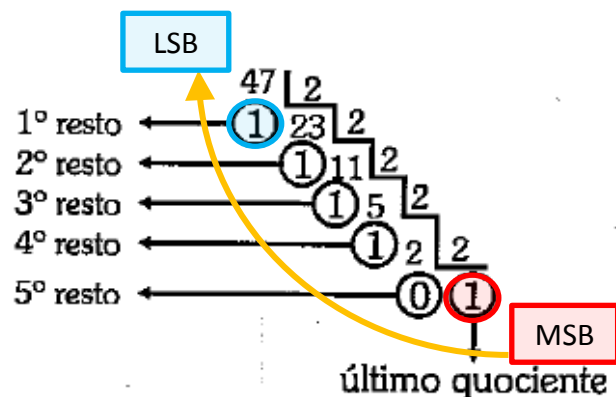


$$\begin{aligned}
 (1011,101)_2 &= (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3}) \\
 &= 8 + 0 + 2 + 1 + 0.5 + 0 + 0.125 \\
 &= 11,625_{10}
 \end{aligned}$$

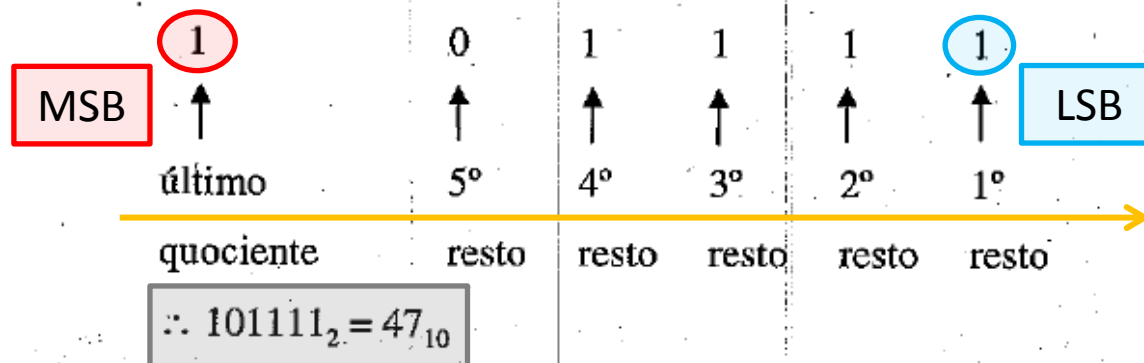
$$(0,0110000010)_2 = (?)_{10}$$

# Conversão Decimal $\rightarrow$ Binário

- Ex: Converter  $(47)_{10}$  para Binário.



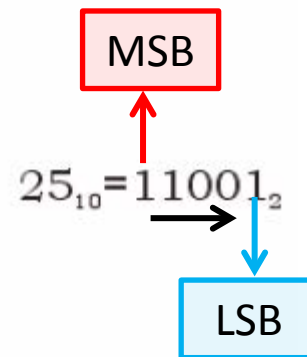
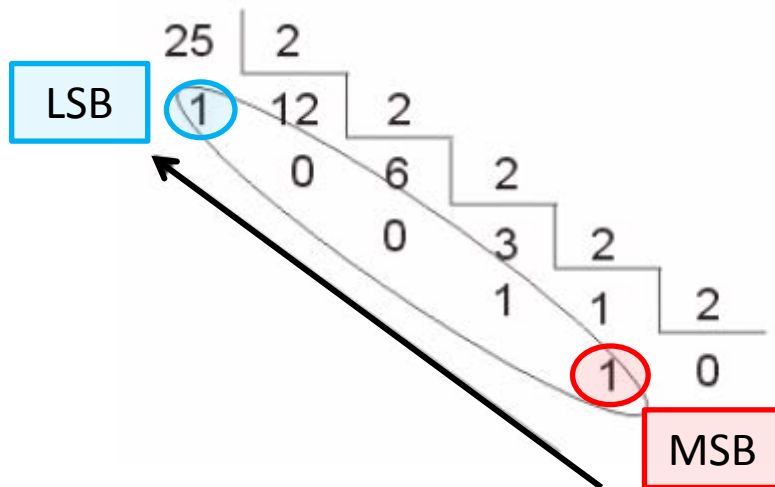
O último quociente será algarismo mais significativo e ficará colocado à esquerda. Os outros algarismos seguem-se na ordem até o 1º resto:





# Conversão Decimal $\rightarrow$ Binário

– Ex: Converter  $(25)_{10}$  para Binário.



– Ex: Decimal:  $(37)_{10}$ ,  
Binário: ?

# Conversão Decimal $\rightarrow$ Binário

- Conversão Fracionária:

– Exemplo: Converter  $(0,828125)_{10} = ( ? )_2$

$$0,828125 \times 2 = 1,65625$$

$$0,65625 \times 2 = 1,3125$$

$$0,3125 \times 2 = 0,625$$

$$0,625 \times 2 = 1,25$$

$$0,25 \times 2 = 0,5$$

$$0,5 \times 2 = 1$$

$$(0,828125)_{10} = (0,110101)_2$$

# Conversão Decimal $\rightarrow$ Binário

- **Conversão Fracionária:**
  - Exemplo: Converter  $(8,375)_{10} = ( ? )_2$

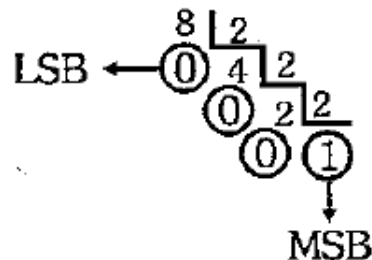
# Conversão Decimal $\rightarrow$ Binário

- **Conversão Fracionária:**

- Exemplo: Converter  $(8,375)_{10} = ( ? )_2$

- $8,375 = 8 + 0,375$

- Parte Inteira



$$\therefore 8_{10} = 1000_2$$

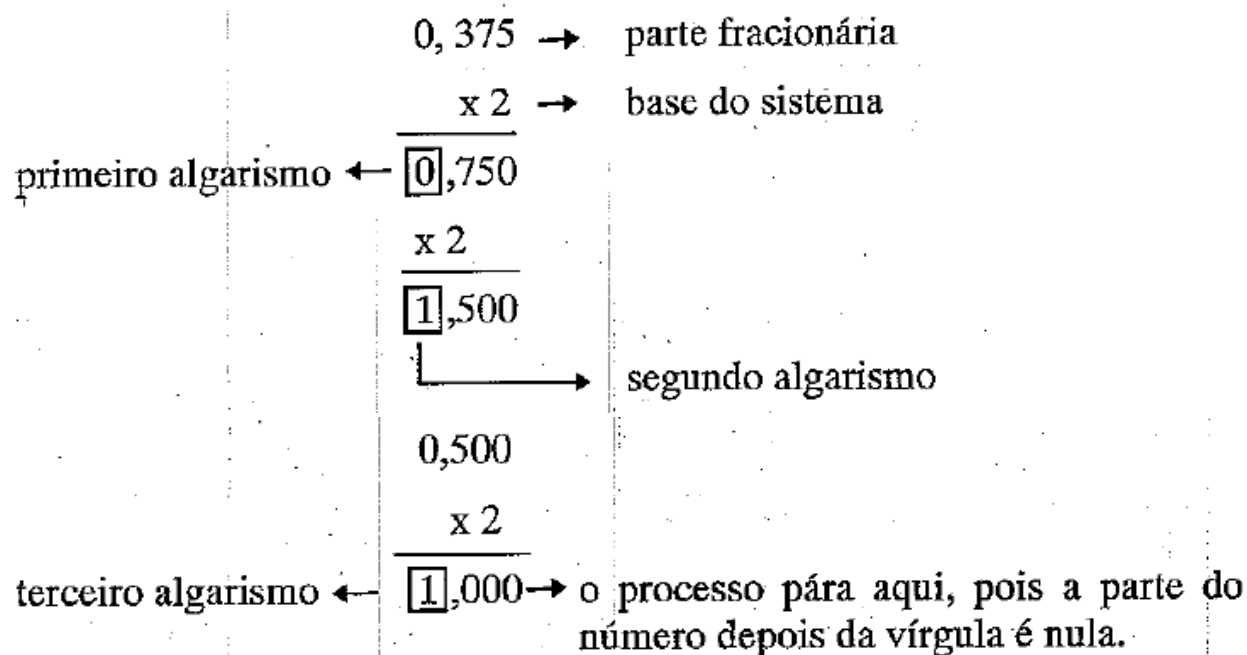
# Conversão Decimal $\rightarrow$ Binário

- Conversão Fracionária:

- Exemplo: Converter  $(8,375)_{10} = ( ? )_2$

- $8,375 = 8 + 0,375$

- Parte Fracionária:



# Conversão Decimal $\rightarrow$ Binário

- Conversão Fracionária:

$$(8,375)_{10} = (\boxed{1000}, \boxed{011})_2$$



# Conversão **Decimal** → **Binário**

## Conversão Fracionária:

- Uma **fração decimal com número finito** de dígitos pode corresponder a uma **fração binária com um número infinito** de dígitos !!!
- Nesses casos, o algoritmo de conversão é suspenso após um **número pré-estabelecido de passos, dependendo da precisão desejada.**
- Portanto, **a precisão** da mudança de base de decimal para binário **depende do número de bits disponíveis** para representar a parte fracionária.



# Representação Binária-Decimal

- Com  $N$  bits podemos contar  $2^N$  diferentes números em decimal (de 0 a  $2^N-1$ ).
- Por exemplo: Qual é a faixa total de valores que podemos representar com 4 bits? Qual o maior valor decimal representado com 4 bits?
  - Para  $N = 4$ , podemos contar de  $0000_2$  até  $1111_2$  ou seja de 0 até 15.
  - Nesse caso, existem  $2^4 = 16$  números decimais diferentes e o maior deles é o  $2^4 - 1 = 15$ .

# Representação Binária-Decimal

- Para representarmos um intervalo de  $M$  valores, necessitamos de  $\lceil \log_2 M \rceil$  bits.
- Para representarmos o número  $X$ , necessitamos no mínimo de:
  - ✓  $(\log_2 X + 1)$  bits, quando  $X$  é múltiplo de potências de 2;
  - ✓  $\lceil \log_2 X \rceil$  bits, para os demais casos.

$$\text{OBS: } \log_2 X = \frac{\log_{10} X}{\log_{10} 2}$$

**Por exemplo:**

1. Quantos bits, no mínimo, são necessários para representar valores decimais na faixa de 0 até 12500? E de 0 até 873 ?
2. Quantos bits, no mínimo, são necessários para representar o número 66 ? E o número 20 ? E o número 75?

# Tabela de Conversão

- ✓ Conversão Binário → Decimal
- ✓ Conversão Decimal → Binário
- Conversão Hexadecimal → Decimal
- Conversão Decimal → Hexadecimal
- Conversão Hexadecimal → Binário
- Conversão Binário → Hexadecimal
- Conversão Octal → Binário
- Conversão Binário → Octal
- Conversão Octal → Decimal
- Conversão Decimal → Octal
- Conversão Octal → Hexadecimal
- Conversão Hexadecimal → Octal

base numérica	decimal	binária	hexadecimal	octal
decimal				
binária				
hexadecimal				
octal				

# Sistema Hexadecimal

- O sistema hexadecimal é um sistema de numeração posicional que representa os números em base 16, portanto empregando 16 símbolos.
- Símbolos: **0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F**
- Está vinculado à informática, pois os computadores costumam utilizar o byte como unidade básica da memória.
- Ele é muito utilizado para representar números binários de uma forma mais compacta, pois é muito fácil converter binários pra hexadecimal e vice-versa.
- Quando manipulamos números com uma extensa quantidade de bits, é mais conveniente, e menos propenso a erros, escrevê-los em hexadecimal.
- Dessa forma, esse sistema é bastante utilizado em aplicações de computadores e microprocessadores.

# Conversão Hexadecimal ↔ Decimal

Decimal	Hexadecimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

# Conversão Hexadecimal → Decimal

- **Conversão Inteira:**
  - Soma dos produtos de cada bit por seu peso relativo.

**Exemplo: 10B2<sub>H</sub>**

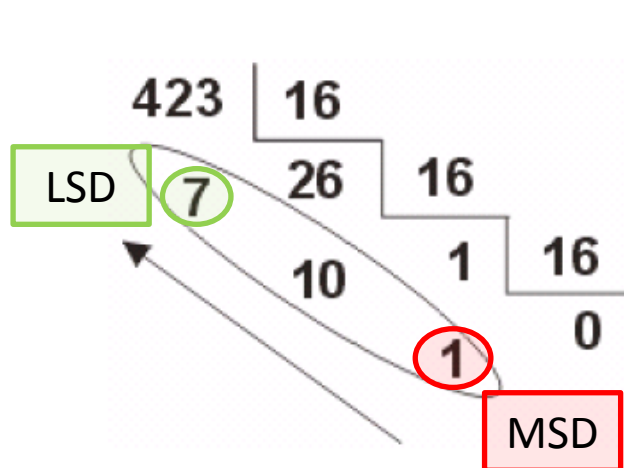
1	0	B	2
$1 \times 16^3$	$0 \times 16^2$	$11 \times 16^1$	$2 \times 16^0$
4.096	0	176	2
$4.096 + 0 + 176 + 2 = 4.274$			

# Conversão **Decimal** → **Hexadecimal**

- Fizemos a conversão de decimal em binário usando divisões sucessivas por 2. Da mesma maneira, a conversão de decimal em hexa pode ser feita usando divisões sucessivas por **16**.
- Cuidado: Se uma calculadora for usada para calcular as divisões no processo de conversão, o resultado incluirá uma fração decimal em vez de um resto.
- O resto pode ser obtido multiplicando-se a parte fracionária por **16**.

# Conversão Decimal $\rightarrow$ Hexadecimal

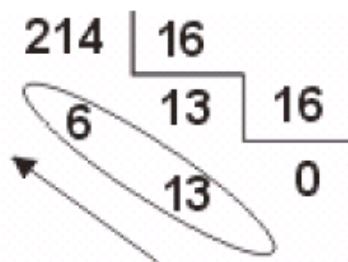
Converter  $423_{10}$  em hexa:



$$423_{10} = 1A7_{16}$$

The diagram shows the hexadecimal result 1A7<sub>16</sub> with a red box labeled 'MSD' above the '1' and a green box labeled 'LSD' below the '7'. A red arrow points from the LSD box to the MSD box, and a green arrow points from the MSD box to the LSD box, indicating the order of reading from bottom to top.

Converter  $214_{10}$  em hexa:



$$214_{10} = D6_{16}$$



# Conversão Decimal $\rightarrow$ Hexadecimal

- Conversão Fracionária:

– Exemplo: Converter  $(0,06640625)_{10} = ( ? )_{16}$

$$\begin{aligned} 0,06640625 \times 16 &= 1,0625 \\ 0,0625 \times 16 &= 1 \end{aligned}$$

$$(0,06640625)_{10} = (0,11)_{16}$$

# Tabela de Conversão

- ✓ Conversão Binário → Decimal
- ✓ Conversão Decimal → Binário
- ✓ Conversão Hexadecimal → Decimal
- ✓ Conversão Decimal → Hexadecimal

- Conversão Hexadecimal → Binário
- Conversão Binário → Hexadecimal

- Conversão Octal → Binário
- Conversão Binário → Octal

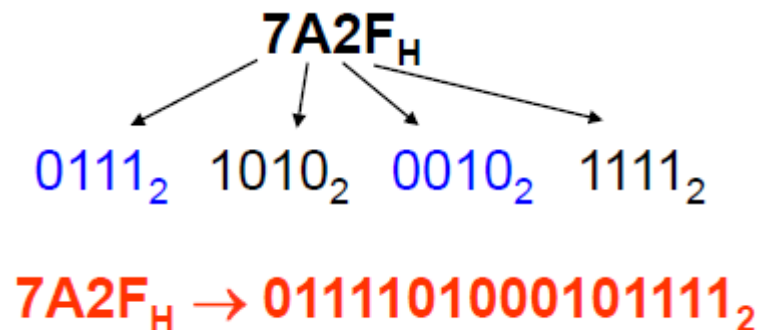
- Conversão Octal → Decimal
- Conversão Decimal → Octal

- Conversão Octal → Hexadecimal
- Conversão Hexadecimal → Octal

base numérica	decimal	binária	hexadecimal	octal
decimal				
binária				
hexadecimal				
octal				

# Conversão Hexadecimal $\rightarrow$ Binário

- Cada algarismo hexa é convertido em seu equivalente binário representado em **4 bits**.
- Para um número de **N** dígitos em hexadecimal, o seu equivalente em binário terá **(4\*N)** bits.



Hexadecimal	Binário
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

# Conversão Hexadecimal → Binário

- Converta para Binário os números em Hexadecimal:

1AD3 =

F0D5 =

ABCDEF =

123456 =

789ABCDEF =

# Conversão Binário → Hexadecimal

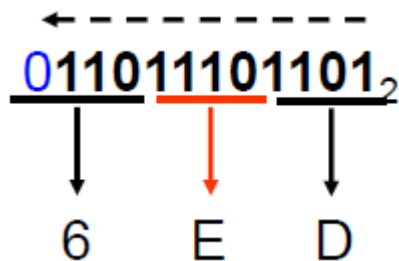
- Divisão dos bits em grupo de 4 e conversão do LSB para o MSB (**direita para esquerda**) de cada grupo no equivalente algarismo hexa.

$$1110100110_2 = \begin{array}{|c|c|c|} \hline 00 & 11 & 1010 & 0110 \\ \hline \downarrow & \downarrow & \downarrow & \downarrow \\ 3 & A & 6 & \end{array}$$

$$= 3A6_{16}$$

Binário	Hexadecimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

# Conversão Binário → Hexadecimal



**11011101101<sub>2</sub> → 6ED<sub>H</sub>**

- $(100101100)_2 = (?)_{16}$

# Conversão Binário → Hexadecimal

## – Conversão Fracionária:

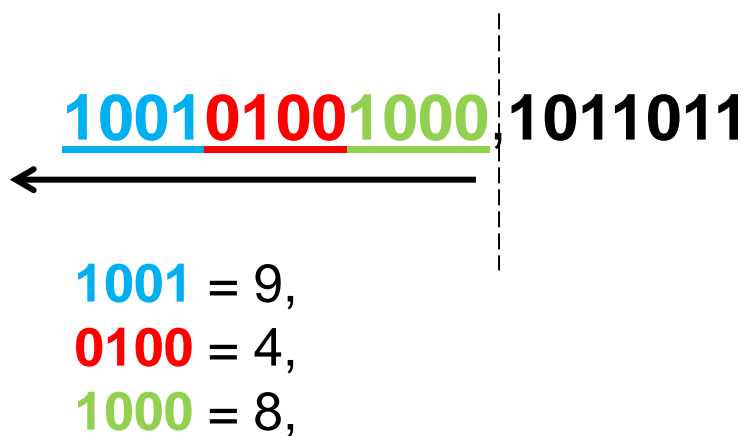
- Agrupa-se o número binário de 4 em 4 dígitos, **da direita para a esquerda na parte inteira** e **da esquerda para a direita na parte fracionária**, e o substitui por seu equivalente hexadecimal

$$(100101001000,1011011)_2 = (?)_{16}$$

# Conversão Binário → Hexadecimal

## – Conversão Fracionária:

- Agrupa-se o número binário de 4 em 4 dígitos, **da direita para a esquerda na parte inteira** e **da esquerda para a direita na parte fracionária**, e o substitui por seu equivalente hexadecimal

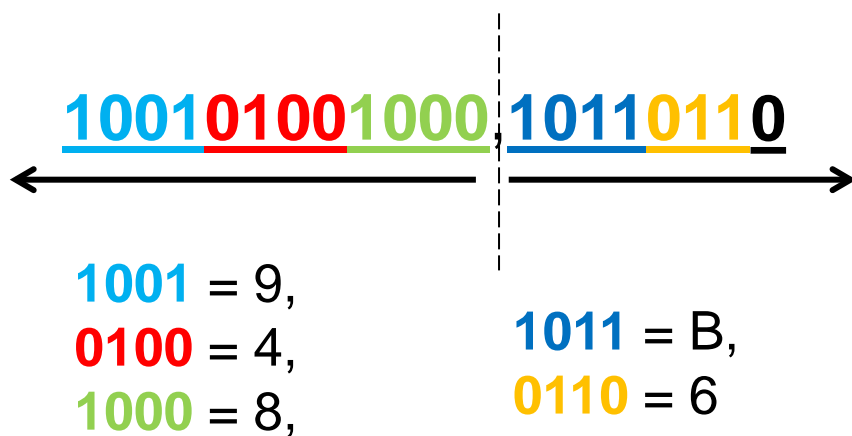




# Conversão Binário → Hexadecimal

## – Conversão Fracionária:

- Agrupa-se o número binário de 4 em 4 dígitos, **da direita para a esquerda na parte inteira** e **da esquerda para a direita na parte fracionária**, e o substitui por seu equivalente hexadecimal



# Conversão Binário → Hexadecimal

## – Conversão Fracionária:

- Agrupa-se o número binário de 4 em 4 dígitos, **da direita para a esquerda na parte inteira** e **da esquerda para a direita na parte fracionária**, e o substitui por seu equivalente hexadecimal

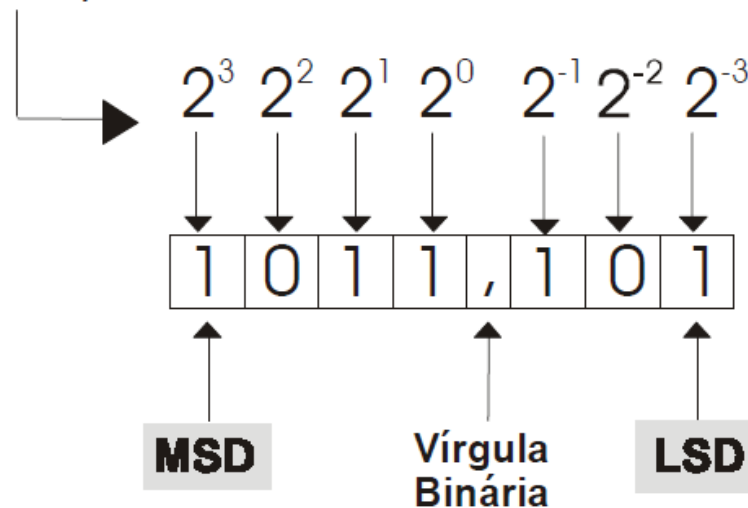
$$(\underline{1001}\underline{0100}\underline{1000}, \underline{1011}\underline{0110})_2 = (948, B6)_{16}$$

# Conversão Binário → Hexadecimal

– Conversão Fracionária:

–  $(1011,101)_2 = (?)_{16}$

Valores Posicionais  
(pesos)



# Tabela de Conversão

- ✓ Conversão Binário → Decimal
- ✓ Conversão Decimal → Binário
- ✓ Conversão Hexadecimal → Decimal
- ✓ Conversão Decimal → Hexadecimal
- ✓ Conversão Hexadecimal → Binário
- ✓ Conversão Binário → Hexadecimal

- Conversão Octal → Binário
- Conversão Binário → Octal
- Conversão Octal → Decimal
- Conversão Decimal → Octal

base numérica	decimal	binária	hexadecimal	octal
decimal				
binária				
hexadecimal				
octal				

- Conversão Octal → Hexadecimal
- Conversão Hexadecimal → Octal

# Contagem em Hexadecimal

- Quando contamos em hexa, cada dígito pode ser incrementado (acrescido de 1) de 0 a F.
- Quando o dígito de uma posição chega ao valor F, este volta a 0, e o dígito da próxima posição é incrementado. Isto é ilustrado nas seguintes sequências de contagem hexa com o respectivo equivalente decimal abaixo.

- HEX: 38 39 3A 3B 68F 69F
- DEC: 56 57 58 59 1679 1695

# Contagem em Hexadecimal

## – Contagem em Hexadecimal

- Com três dígitos hexa podemos contar de  $000_{16}$  a  $FFF_{16}$ , que corresponde à faixa de  $0_{10}$  a  $4095_{10}$  valores diferentes.

## – Vantagens da Base Hexadecimal

- Forma “compacta” de representar sequência de bits;
- Sequências binárias nem sempre representam valores numéricos: podem ser algum tipo de código;
- Conveniente, ao manipular extensas cadeias de bits.

# Conversão Indireta

- Converta o número decimal 378 em um número binário de 16 bits:

$$\underline{378} = 23 + \text{resto } 10 = \textcolor{red}{A}$$

16

$$\underline{23} = 1 + \text{resto } 7 = \textcolor{blue}{7}$$

16

$$\underline{1} = 0 + \text{resto } 1 = \textcolor{green}{1}$$

16

$$378_{10} = \textcolor{green}{1}\textcolor{blue}{7}\textcolor{red}{A}_H = \textcolor{green}{0001}\textcolor{blue}{0111}\textcolor{red}{1010} - 12 \text{ bits}$$

# Conversão Indireta

- Converta o número decimal 378 em um número binário de 16 bits:

$$\underline{378} = 23 + \text{resto } 10 = A$$

16

$$\underline{23} = 1 + \text{resto } 7 = 7$$

16

$$\underline{1} = 0 + \text{resto } 1 = 1$$

16

$$378_{10} = 17A_H = 0000000101111010 - 16 \text{ bits}$$



# Base Octal

- O sistema octal é um sistema de numeração posicional que representa os números em base 8, portanto empregando 8 símbolos.
- Símbolos: 0, 1, 2, 3, 4, 5, 6, 7

Decimal	Octal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	10
9	11
10	12
11	13
12	14
13	15
14	16
15	17

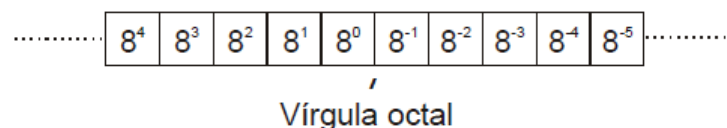
## Conversão Octal $\rightarrow$ Decimal

$$(144)_8 = (?)_{10}$$

$8^2$	$8^1$	$8^0$
1	4	4

$$1 \times 8^2 + 4 \times 8^1 + 4 \times 8^0 =$$

$$\therefore 144_8 = 100_{10}$$



## Conversão Decimal $\rightarrow$ Octal

$$(92)_{10} = (?)_8$$

	92	8	
1º resto $\leftarrow$	4	11	8
2º resto $\leftarrow$	3	1	
último quociente $\leftarrow$			

$$\therefore 92_{10} = 134_8$$

## Conversão Octal $\rightarrow$ Binário

$$(44675)_8 = (?)_2$$

$$\begin{array}{ccc|cc} \overbrace{100}^4 & \overbrace{100}^4 & \overbrace{110}^6 & \overbrace{111}^7 & \overbrace{101}^5 \end{array} \quad \therefore 44675_8 = 100100110111101_2$$

## Conversão Binário $\rightarrow$ Octal

$$(1000110011)_2 = (?)_8$$

$$\begin{array}{cccc} \overbrace{001}^1 & \overbrace{000}^0 & \overbrace{110}^6 & \overbrace{011}^3 \end{array} \quad \therefore 1000110011_2 = 1063_8$$

Binário	Octal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	10
1001	11
1010	12
1011	13
1100	14
1101	15
1110	16
1111	17

# Conversão Octal $\rightarrow$ Hexadecimal

Neste caso é necessário um passo intermediário:

*Hexadecimal*  $\rightarrow$  *Binário*  $\rightarrow$  *Octal*

Exemplo:  $(1F4)_{16} = (?)_8$

**De Hexa para Binário:**  $(1F4)_{16}$

1 = **0001**, F = **1111**, 4 = **0100**

logo:  $(1F4)_{16} = (1\ 1111\ 0100)_2$

**De Binário para Octal:**  $(111\ 110\ 100)_2$

Da direita para a esquerda: 100 = **4**, 110 = **6**, 111 = **7**

logo:  $(111110100)_2 = (764)_8$

Assim:  $(1F4)_{16} = (764)_8$

Hexadecimal	Octal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	10
9	11
A	12
B	13
C	14
D	15
E	16
F	17

# Conversão Hexadecimal $\rightarrow$ Octal

O mesmo acontece neste caso:

*Octal  $\rightarrow$  Binário  $\rightarrow$  Hexadecimal*

Exemplo:  $(144)_8 = (?)_{16}$

**De Octal para Binário:  $(144)_8$**

**1 = 001, 4 = 100, 4 = 100**

logo:  $(144)_8 = (001\ 100\ 100)_2$

**De Binário para Octal:  $(0\ 0110\ 0100)_2$**

Da direita para a esquerda: 0100 = **4**, 0110 = **6**, 0000 = **0**

logo:  $(001100100)_2 = (\mathbf{64})_{16}$

Assim:  $(144)_8 = (\mathbf{64})_{16}$

Hexadecimal	Octal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	10
9	11
A	12
B	13
C	14
D	15
E	16
F	17

# Tabela de Conversão

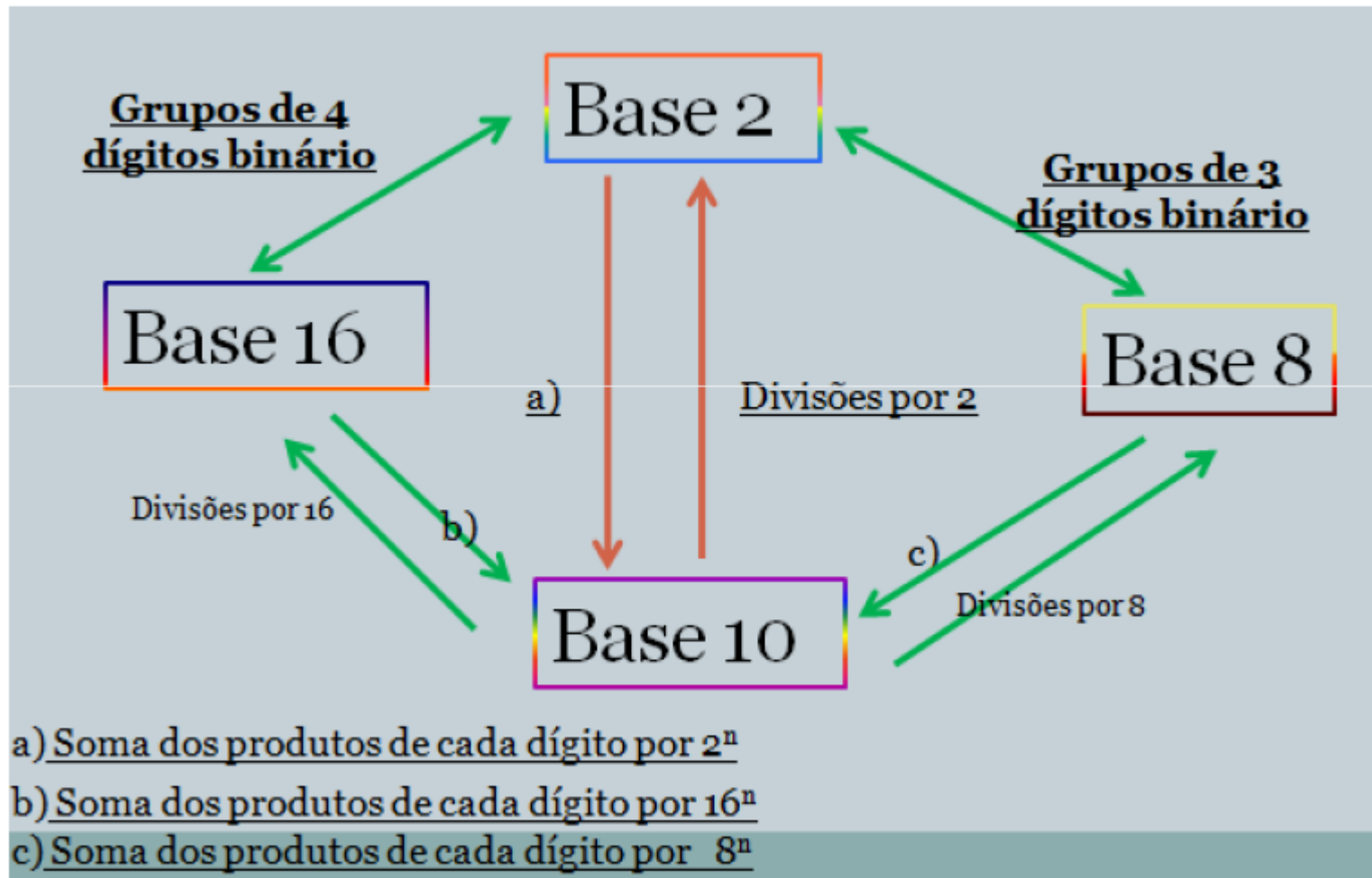
- ✓ Conversão Binário → Decimal
- ✓ Conversão Decimal → Binário
  
- ✓ Conversão Hexadecimal → Decimal
- ✓ Conversão Decimal → Hexadecimal
  
- ✓ Conversão Hexadecimal → Binário
- ✓ Conversão Binário → Hexadecimal
  
- ✓ Conversão Octal → Binário
- ✓ Conversão Binário → Octal
  
- ✓ Conversão Octal → Decimal
- ✓ Conversão Decimal → Octal
  
- ✓ Conversão Octal → Hexadecimal
- ✓ Conversão Hexadecimal → Octal

base numérica	decimal	binária	hexadecimal	octal
decimal				
binária				
hexadecimal				
octal				

# Resumo das Conversão de Bases Numéricas

Decimal	Binário	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

# Resumo Conversões de Bases Numéricas



[http://coolconversion.com/math/binary-octal-hexa-decimal/\\_binary\\_\\_11010111\\_\\_decimal\\_](http://coolconversion.com/math/binary-octal-hexa-decimal/_binary__11010111__decimal_)



# Exercícios

## Exercícios

- 1) Converta  $24CE_{16}$  para decimal.
- 2) Converta  $3117_{10}$  para hexa e depois para binário.
- 3) Converta  $1001011110110101_2$  para hexa.
- 4) Encontre os quatro números seguintes da sequência hexa: E9A, E9B, E9C, E9D, \_\_\_\_\_, \_\_\_\_\_, \_\_\_\_\_.
- 5) Converta  $3527_8$  para hexa.

## Mais exercícios

- 1) Converta os seguintes números binários em decimal:

- |                 |               |
|-----------------|---------------|
| a) 10110        | d) 1111010111 |
| b) 10001101     | e) 10111111   |
| c) 100100001001 |               |

- 2) Converta os seguintes valores decimais em binário:

- |        |         |
|--------|---------|
| a) 37  | d) 205  |
| b) 14  | e) 2313 |
| c) 189 | f) 511  |

- 3) Qual o maior número decimal que pode ser representado por um número binário de oito bits ? E de 16 bits ?

- 4) Converta cada número octal em seu equivalente decimal:

- |         |         |
|---------|---------|
| a) 743  | d) 257  |
| b) 36   | e) 1204 |
| c) 3777 |         |

- 5) Converta cada número decimal em binário:

- |        |          |
|--------|----------|
| a) 59  | c) 65535 |
| b) 372 | d) 255   |

- 6) Converta cada número octal do item 4 em binário:

- 7) Converta cada número binário do item 1 em octal:

- 8) Liste todos os números octais entre  $165_8$  e  $200_8$ .

- 9) Converta os seguintes números hexa em decimal:

- |         |        |
|---------|--------|
| a) 92   | d) 2C0 |
| b) 1A6  | e) 7FF |
| c) 37FD |        |

- 10) Converta os seguintes números decimais em hexa:

- |         |          |
|---------|----------|
| a) 75   | d) 25619 |
| b) 314  | e) 4095  |
| c) 2048 |          |

- 11) Converta os números binários do item 1 em hexa.

- 12) Converta os números hexa do item 10 em binário.

13) Na maioria dos microcomputadores o endereço das células de memória é hexadecimal. tais endereços são números sequenciais que identificam cada posição de memória.

a) Um determinado microcomputador pode armazenar números de oito bits em cada célula de memória. Sabendo-se que a faixa de endereçamento vai de  $0000_{16}$  até  $FFFF_{16}$ , quantas células existem nesta memória ?

b) Outro microcomputador tem 4096 células. Qual a faixa de endereçamento em hexadecimal desta memória ?

- 14) Liste sequencialmente, em hexadecimal, os números de  $280_{16}$  até  $2A0_{16}$ .

- 15) execute as conversões abaixo:

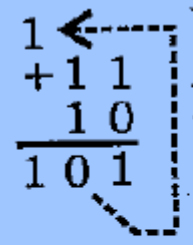
- |   |                                       |
|---|---------------------------------------|
| a) $1417_{10} =$ _____ <sub>2</sub>     | g) $2358 =$ _____ <sub>10</sub>       |
| b) $255_{10} =$ _____ <sub>2</sub>      | h) $43168 =$ _____ <sub>10</sub>      |
| c) $110100012 =$ _____ <sub>10</sub>    | i) $7A916 =$ _____ <sub>10</sub>      |
| d) $111010100012 =$ _____ <sub>10</sub> | j) $3E1C16 =$ _____ <sub>10</sub>     |
| e) $2497_{10} =$ _____ <sub>8</sub>     | k) $1600_{10} =$ _____ <sub>16</sub>  |
| f) $511_{10} =$ _____ <sub>8</sub>      | l) $38187_{10} =$ _____ <sub>16</sub> |

# Operações Aritméticas no Sistema Binário

# Operações Aritméticas no Sistema Binário

– **ADIÇÃO**: Regras Básicas para Adição em Binário:

- $0 + 0 = 0$
- $0 + 1 = 1$
- $1 + 0 = 1$
- $1 + 1 = 0$  (Transporta 1 = Carry Out = Vai Um)



Handwritten binary addition example:

$$\begin{array}{r} 1 \leftarrow \\ + 11 \\ 10 \\ \hline 101 \end{array}$$

1 + 1 = 0 e transporta 1

$\therefore 11_2 + 10_2 = 101_2$

Verificação:  $(3_{10} + 2_{10} = 5_{10})$

$$\begin{array}{r} 11010 \\ + 10010 \\ \hline \end{array}$$

# Operações Aritméticas no Sistema Binário

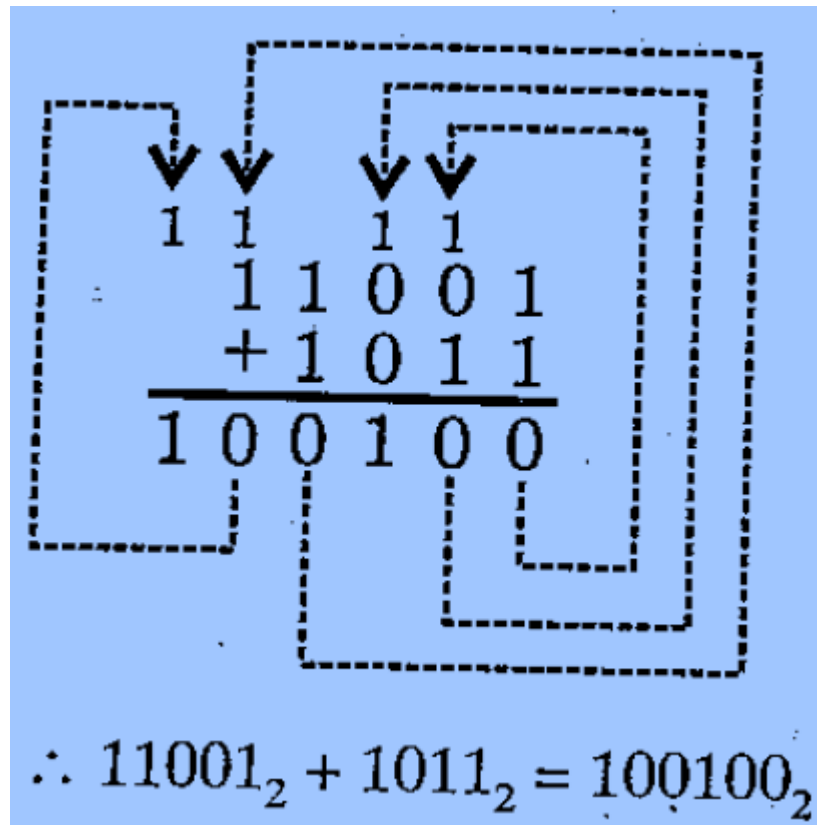
– ADIÇÃO: Regras Básicas para Adição em Binário:

- $0 + 0 = 0$
- $0 + 1 = 1$
- $1 + 0 = 1$
- $1 + 1 = 0$  ( Transporta 1 = Carry Out = Vai Um)

$$\begin{array}{r} \textcolor{red}{1} \\ 11010 \\ + 10010 \\ \hline \textcolor{red}{1}01100 \end{array}$$

## Adição: Exemplo

$$11001_2 + 1011_2:$$



# Operações Aritméticas no Sistema Binário

– **DIFERENÇA**: Regras Básicas para Diferença em Binário:

- $0 - 0 = 0$
- $1 - 1 = 0$
- $1 - 0 = 1$
- $0 - 1 = 1$  ( Empresta 1 )

**Ex:  $1000 - 111 = ?$**

$$\begin{array}{r}
 1000 \\
 -111 \\
 \hline
 001
 \end{array}$$

0-1=1 e transporta 1 para a coluna seguinte

$$\begin{array}{r}
 1000 \\
 -111 \\
 \hline
 01
 \end{array}$$

transporte anterior  
0-1-1=0 e transporta 1 para a coluna seguinte

$$\begin{array}{r}
 1000 \\
 -111 \\
 \hline
 001
 \end{array}$$

0-1-1=0 e transporta 1 para a coluna seguinte

$$\begin{array}{r}
 1000 \\
 -111 \\
 \hline
 0001
 \end{array}$$

transporte anterior  
1-1=0

$$\therefore 1000_2 - 111_2 = 0001_2$$

## Diferença: Exemplo

$$\begin{array}{r} 11010 \\ - 10011 \\ \hline \end{array}$$

$$\begin{array}{r} 11010 \\ \phantom{0}^1\phantom{0}^1\phantom{0}^1 \\ - 10011 \\ \hline 00111 \end{array}$$

# Operações Aritméticas no Sistema Binário

– **MULTIPLICAÇÃO**: Regras Básicas para Multiplicação em Binário:

- $0 * 0 = 0$
- $0 * 1 = 0$
- $1 * 0 = 0$
- $1 * 1 = 1$

Ex: 
$$\begin{array}{r} 11 \\ * 101 \\ \hline \end{array}$$



# Operações Aritméticas no Sistema Binário

– **MULTIPLICAÇÃO**: Regras Básicas para Multiplicação em Binário:

- $0 * 0 = 0$
- $0 * 1 = 0$
- $1 * 0 = 0$
- $1 * 1 = 1$

Ex:

$$\begin{array}{r} 11 \\ * 101 \\ \hline 11 \\ 00 \\ 11 \\ \hline 1111 \end{array}$$

# Operações Aritméticas no Sistema Binário

- Generalizando a **MULTIPLICAÇÃO...**

$$\begin{array}{cccccccc}
 & & & & x_7y_0 & x_6y_0 & x_5y_0 & x_4y_0 & x_3y_0 & x_2y_0 & x_1y_0 & x_0y_0 \\
 & & & & x_7y_1 & x_6y_1 & x_5y_1 & x_4y_1 & x_3y_1 & x_2y_1 & x_1y_1 & x_0y_1 \\
 & & & & & & & & & & & \\
 & & & & x_7y_2 & x_6y_2 & x_5y_2 & x_4y_2 & x_3y_2 & x_2y_2 & x_1y_2 & x_0y_2 \\
 & & & & & & & & & & & \\
 & & & & x_7y_3 & x_6y_3 & x_5y_3 & x_4y_3 & x_3y_3 & x_2y_3 & x_1y_3 & x_0y_3 \\
 & & & & & & & & & & & \\
 & & & & x_7y_4 & x_6y_4 & x_5y_4 & x_4y_4 & x_3y_4 & x_2y_4 & x_1y_4 & x_0y_4 \\
 & & & & & & & & & & & \\
 & & & & x_7y_5 & x_6y_5 & x_5y_5 & x_4y_5 & x_3y_5 & x_2y_5 & x_1y_5 & x_0y_5
 \end{array}$$

Multiplier implementation:

- $m$  arrays of  $n$  AND gates
- $m - 1$   $n$ -bit adders

# Operações Aritméticas no Sistema Binário

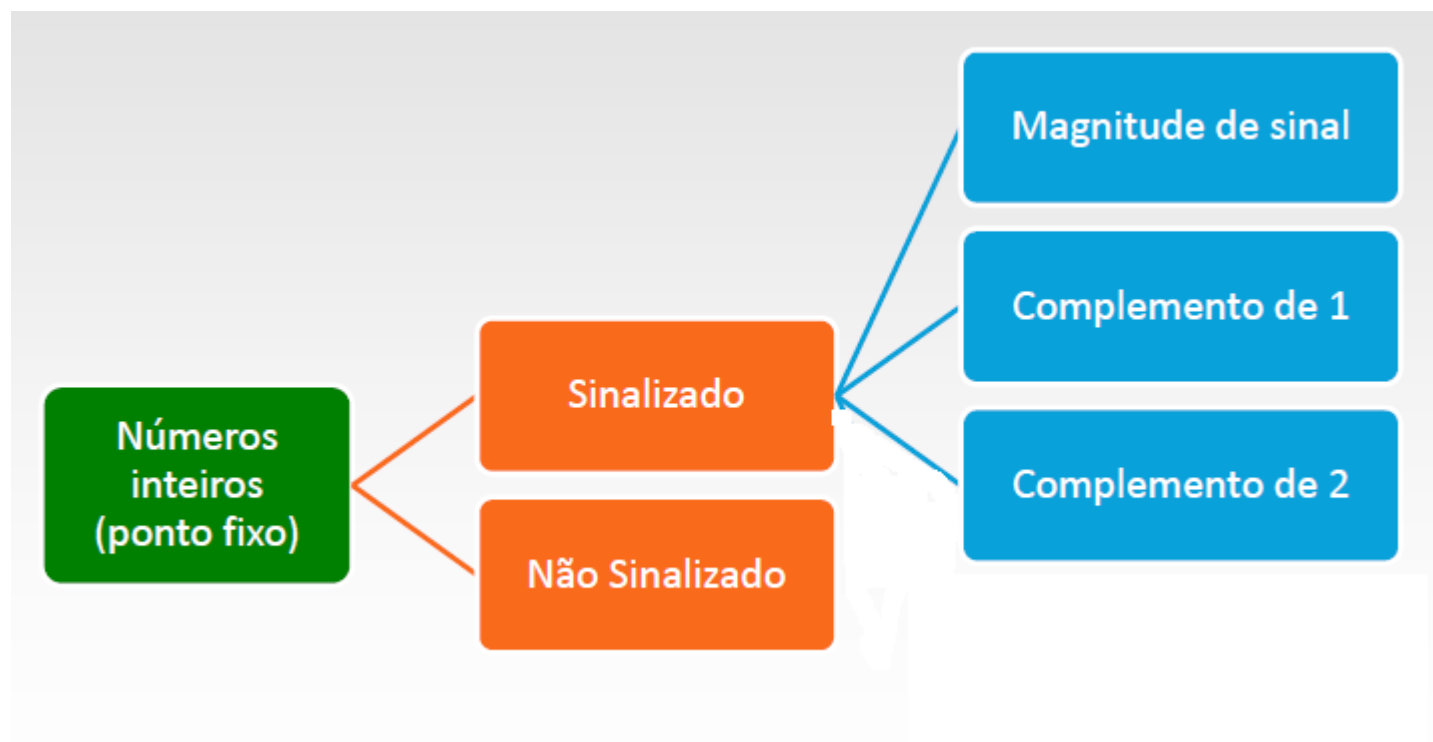
Utiliza o mesmo método que a operação “divisão” no sistema decimal: deslocamentos e subtrações.

$$\begin{array}{r} 42 \overline{) 6} \\ 7 \end{array}$$

$$\begin{array}{r} 101010 \overline{) 110} \\ -110 \\ \hline 1001 \\ -110 \\ \hline 0110 \\ 110 \\ \hline 000 \end{array}$$

# Representações de Números Negativos

# Números Inteiros



# Sinal de Magnitude ou Sinal de Módulo

- Na base 10 (podendo utilizar em outras bases) a representação de um número positivo é feita apresentando somente o número, ou o número com um sinal(+), à esquerda, já os negativos possuem um sinal(-) à esquerda.
- Como adicionar um símbolo a um número binário se um computador "entende" somente 1 e 0 ?
- Assim, para a forma de representação de sinal e magnitude em um número binário de 8 dígitos, o **MSB (primeiro dígito da esquerda para a direita) deve ser considerado o sinal - Bit de Sinal.**
  - 0 sinal **positivo**. Exemplo: 0000 0011 = 3
  - 1 o sinal é **negativo**. Exemplo: 1000 0011 = -3
- Em um sistema de sinal-magnitude de **8 bits**, temos **7 bits para representar o valor e 1 bit (MSB) para representar o sinal.**

# Sinal de Magnitude ou Sinal de Módulo

**Ex 1)**  $35_{10} = 10\ 0011_2$  (7 bits)

$$+ \underline{10\ 0011}_2 = 0 \underline{10\ 0011}_2$$

→ Bit de sinal (0 → indica número positivo)

**Ex 2)**  $73_{10} = 100\ 1001_2$  (8 bits)

$$- \underline{100\ 1001}_2 = 1 \underline{100\ 1001}_2$$

→ Bit de sinal (1 → indica número negativo)

**OBS:** O Problema que o Zero acaba tendo duas representações:

$$0000\ 0000 = +0$$

$$1000\ 0000 = -0$$

A faixa (*range*) de valores vai de  $-(2^{n-1} - 1)$  até  $+(2^{n-1} - 1)$ , e 2 possibilidades de zero.

## Complemento a (Base-1)

- **Complemento** é a diferença entre o maior algarismo possível na base e cada algarismo do número.
- A representação dos **números inteiros negativos** é obtida efetuando-se: **(base - 1) menos cada algarismo do número**. Fica mais fácil entender através de exemplos...
- Veremos mais detalhes sobre o **Complemento de 1** na sequência.



# Complemento a (Base-1)

Ex 1: Calcular o complemento a (base - 1) do número  $297_{10}$

Se a base é 10, então  $10 - 1 = 9$  e o complemento a (base -1) será complemento a 9.

$$\begin{array}{r}
 \text{Ex.1} \\
 \text{(base -1) ---> } 999 \\
 \quad \quad \quad - \underline{297} \\
 \text{Complemento ---> } 702
 \end{array}$$

Ex 2: Calcular o complemento a (base - 1) do número  $3A7E_{16}$

Se a base é 16, então  $16 - 1 = 15 = F$  e o complemento a (base -1) será complemento a F.

$$\begin{array}{r}
 \text{Ex.2} \\
 \text{(base -1) ---> } FFFF \\
 \quad \quad \quad - \underline{3A7E} \\
 \text{Complemento ---> } C581
 \end{array}$$

# Complemento a Base

- A representação dos **números inteiros negativos** em complemento a base é obtida **calculando-se o complemento a (base -1) e depois somando 1 ao resultado**.
- Veremos mais detalhes sobre o **Complemento de 2** na sequência.

Ex 1: calcular o complemento a base do número  $297_{10}$

Queremos então calcular o complemento a 10 (C10) desse número.

<u>Ex.1</u>	<u>Ex.1 (alternativa)</u>
1000	999
- <u>297</u>	- <u>297</u>
703	702
	+ <u>001</u>
	703

Note que o método alternativo é mais eficiente.

Ex 2: calcular o complemento a base do número  $3A7E_{16}$

Queremos então calcular o complemento a 16 (C16) desse número.

<u>Ex.2</u>
FFFF
- <u>3A7E</u>
C581
+ <u>0001</u>
C582

# Complemento de 1

- **Complemento de 1:  $C_1(X)$**

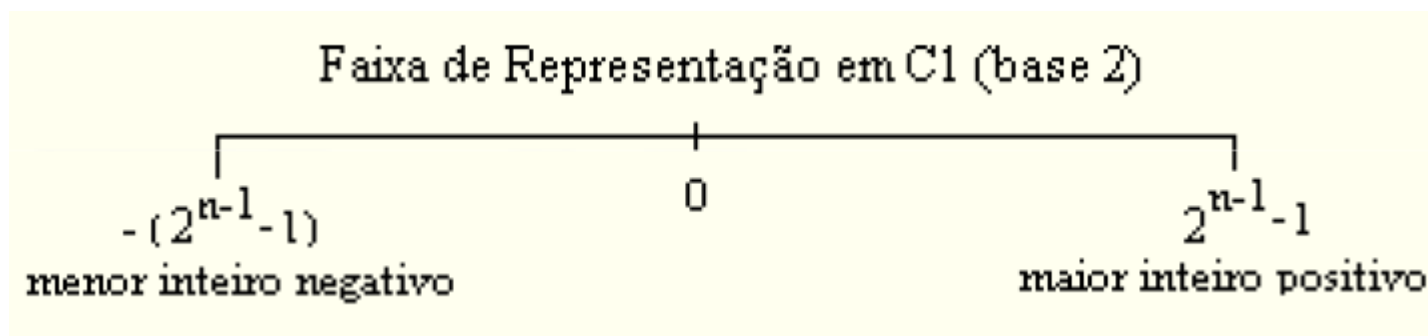
- Outra maneira de representar **números binários negativos** consiste em inverter todos os bits, ou seja, substituindo os **0's por 1's** e os **1's por 0's**.
- **Os números positivos permanecem inalterados**, sendo o primeiro dígito o 0. Nos números negativos o primeiro dígito é o 1. Podendo representar o zero, 00000000 e **11111111**.
- Para se obter o complemento a 1 de um número binário, devemos **subtrair cada algarismo de 1**. Uma particularidade dos números binários é que, para efetuar esta operação, **basta inverter todos os bits**.

EX: 0010 1011 (43)  
 (-43) = 1111 1111 - 0010 1011  
 = 1101 0100 (-43)

$$C_1(X) = \bar{X} = (1)_{nbits} - X_{nbits}$$

# Complemento de 1

Decimal (positivo)	Binário (se o número é positivo, não há alteração)	Decimal (negativo)	Binário (em C1)
0	0000	0	1111
1	0001	-1	1110
2	0010	-2	1101
3	0011	-3	1100
4	0100	-4	1011
5	0101	-5	1010
6	0110	-6	1001
7	0111	-7	1000



# Complemento de 1

- Atenção na operação de Complemento de 1 quando tiver o vai um na última coluna.

binário	decimal	
11111110	-1	
+ 00000010	+2	
.....	...	
1 00000000	0	<-- não é a resposta correta
1	+1	<-- adiciono "vai-um"
.....	...	
00000001	1	<-- resposta correta

# Complemento de 2

- Os problemas de múltiplas representações do 0 e a necessidade de tratamento com "vai-um" (exemplo anterior) são contornadas por um sistema chamado de complemento de 2.
- **O bit mais significativo (MSB) indica o sinal do número.**
  - **0 Positivo**
  - **1 Negativo**
- Os **números positivos** são representados **normalmente em binário** em complemento 2.
  - Exemplo:  $(90)_{10} = (0101\ 1010)_2$
- Já os **números negativos** são representados da seguinte forma:
  - **Converta para binário**, de maneira normal, o número como se não tivesse sinal.
  - **Inverta todos os bits**, trocando 1 por 0 e 0 por 1 para obter o complemento 1.
  - **Some 1** ao número obtido no passo 2 para obter o complemento 2.

# Complemento de 2

$$C_2(X) = C_1(X) + 1$$

➤ **Exemplo:** Converta o número **-39** para a forma complemento a 2.

**1º passo:** **Converta** somente o número 39 (esqueça o sinal) **para binário** (8 bits) e obtém-se 0010 0111.

**2º passo:** **Inverta todos os bits** e obtém-se 1101 1000 que é o complemento 1.

**3º passo:** **Somando 1** a 1101 1000 obtém-se **1101 1001** que é -39 em complemento 2.

# Complemento de 2

## ➤ Converter de COMPLEMENTO DE 2 PARA DECIMAL.

- Se o número for **positivo** utiliza a forma normal !
- Se o número for **negativo** utilize a seguinte regra:

- Some os pesos dos bits 1.
- Considerando o peso do MSB como negativo.

### • Exemplo:

$$(-2^7) \text{ } 2^6 \text{ } 2^5 \text{ } 2^4 \text{ } 2^3 \text{ } 2^2 \text{ } 2^1 \text{ } 2^0$$

$$\underline{1}1011001 = (64+16+8+1-128) = 89-128 = -39$$

❖ Note que:

0	$a_{n-2}$	...	$a_2$	$a_1$	$a_0$	➔	$A = -0 \cdot 2^{n-1} + \sum_{i=0}^{n-2} a_i \cdot 2^i$
1	$a_{n-2}$	...	$a_2$	$a_1$	$a_0$	➔	$A = -1 \cdot 2^{n-1} + \sum_{i=0}^{n-2} a_i \cdot 2^i$



# Complemento de 2

- Converter de COMPLEMENTO DE 2 PARA BINÁRIO/DECIMAL.

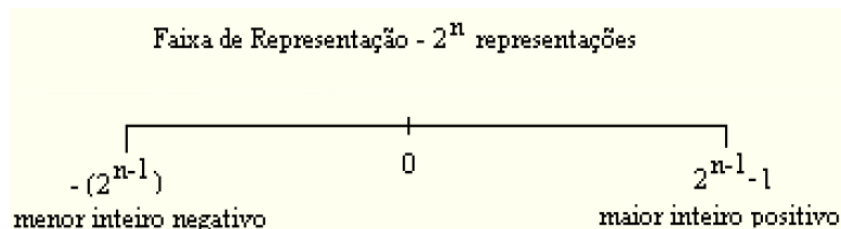
OU...

- **Aplica-se novamente o C2** → binário.
- Binário → decimal

- Exemplo anterior:

$$(1101\ 1001)_{C2} \rightarrow 0010\ 0110 + 1 = (-0010\ 0111)_2 \rightarrow (-39)_{10}$$

# Complemento de 2 vs Binário vs Decimal

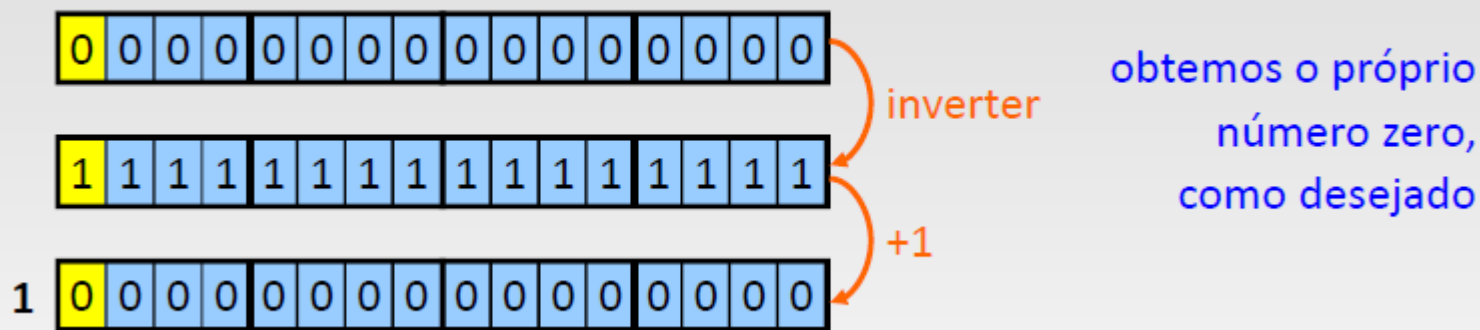


**OBS!!** Um mesmo símbolo hexadecimal (ou binário) pode corresponder a valores decimais diferentes (por exemplo:  $A = -6 = 10$ ). Por isso, **é necessário explicitar qual das duas representações está sendo usada**. Ou seja,  $A = -6$  se for complemento de dois (ou seja, número inteiro) e  $A = 10$  se não for complemento de dois (ou seja, número natural).

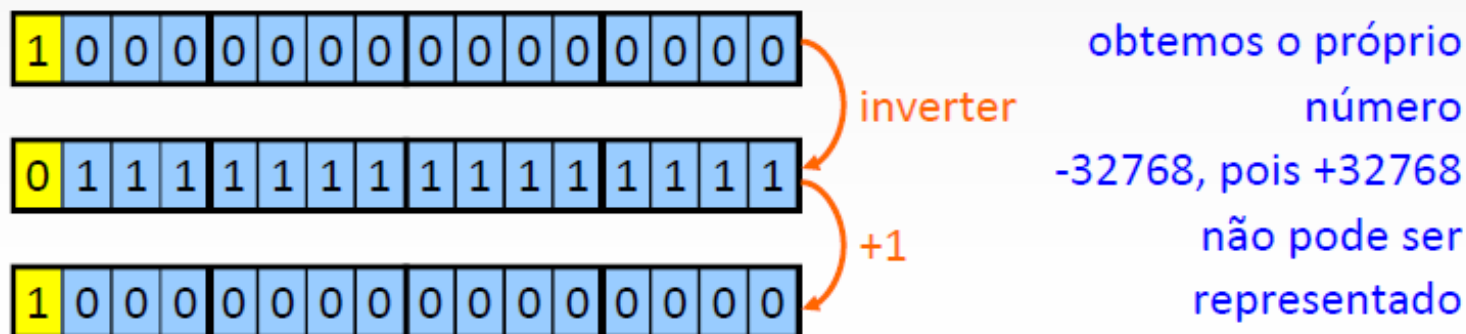
Binário	Natural	Inteiro	Hexadecimal
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	8	-8	8
1001	9	-7	9
1010	10	-6	A
1011	11	-5	B
1100	12	-4	C
1101	13	-3	D
1110	14	-2	E
1111	15	-1	F

# Casos Especiais (Complemento de 2)

❖ Quanto vale  $-0$ ?



❖ Quanto vale  $-(-32768 = 2^{15})$ ?



# Complemento de 2 em Operações Aritméticas

## ➤ SUBTRAÇÃO:

$$\bullet A - B = A + (-B) = A + (\bar{B} + 1)$$

$\rightarrow C_2(B)$

$$\begin{array}{r} 1101\ 0111 - \\ 0010\ 0101 \\ \hline = ?\ (8\ bits) \end{array}$$

$$215 - 37 =$$

$$\begin{array}{l} = 178 \\ = (1011\ 0010)_2 \end{array}$$

$11010111_2 - 100101_2$ Complemento de 1 de 00100101: 11011010 Complemento de 2:  Operação:	$\begin{array}{r} 11011010 \\ + \quad 1 \\ \hline 11011011 \\ 11010111 \\ + 11011011 \\ \hline \text{X}10110010 \end{array}$ <p><math>\rightarrow</math> estouro do número de bits desconsiderado</p>
---	---

$\therefore 11010111_2 - 100101_2 = 10110010_2$

**OVERFLOW ou ESTOURO:** número de bits extrapola o máximo usado (ou seja, o resultado excede o limite de representação). Neste caso é desconsiderado.

# Exemplo

$$10011_2 - 100101_2$$

Trata-se de um número menor subtraindo um outro maior. Agindo da mesma forma, temos:

Complemento de 1 de 100101: 011010

Complemento de 2: 011011

Operação: 010011

+011011

101110

Pelo fato de o minuendo ( $10011_2$ ) ser menor que o subtraendo ( $100101_2$ ) a resposta é negativa, estando na notação do complemento de 2. Para obtê-la na notação binária normal, basta determinar novamente o complemento de 2 e acrescentar o sinal negativo:

$$101110 \Rightarrow 010001 \Rightarrow 010001 + 1 = 010010$$

$$\therefore 10011_2 - 100101_2 = -10010_2 \text{ (ou } 101110 \text{ em complemento de 2)}$$

Se o número é **positivo**,  
mantenha-o;

Se o número é **negativo**,  
complemente-o;

E aí, é só somar.

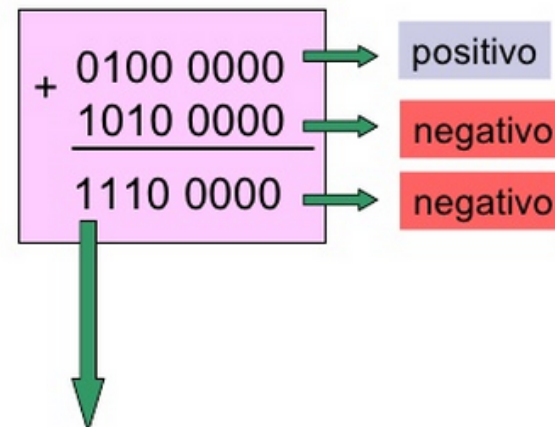
# Detecção de Overflow

- Via de regra, a adição de dois números com **sinais diferentes nunca produz um overflow.**
- *O overflow* aparecerá sempre que o **sinal do resultado da soma é diferente do sinal dos operandos, quando iguais**
  - Some os dois números e observe se ocorre o *carry* (vai 1) **sobre** o bit de sinal e se ocorre o *carry* **após** o bit de sinal.
  - Se ocorrer **um e somente um** dos dois *carry*, então **houve estouro**; caso contrário o resultado da soma está dentro do campo de definição.

# Detecção de Overflow

- Some os dois números e observe se ocorre o *carry* (vai 1) **sobre** o bit de sinal e se ocorre o *carry* **após** o bit de sinal.
- Se ocorrer **um e somente um** dos dois *carry*, então **houve estouro**; caso contrário o resultado da soma está dentro do campo de definição.

Adição de operandos com sinais opostos (8 bits)

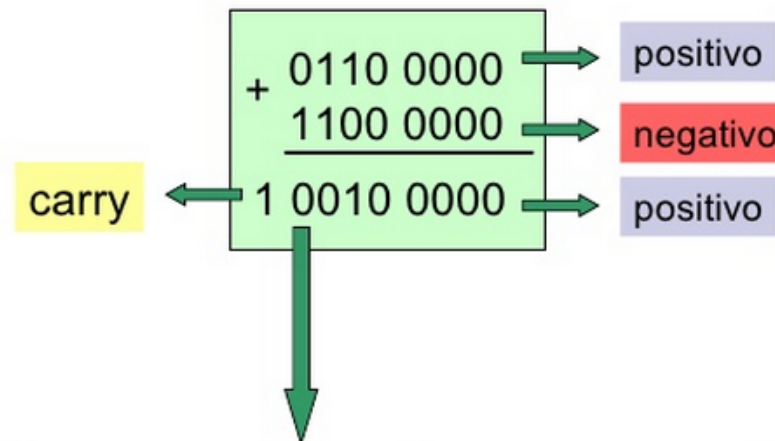


- Não ocorre overflow, o resultado é negativo e está em complemento a 2

# Detecção de Overflow

- Some os dois números e observe se ocorre o *carry* (vai 1) **sobre** o bit de sinal e se ocorre o *carry* **após** o bit de sinal.
- Se ocorrer **um e somente um** dos dois *carry*, então **houve estouro**; caso contrário o resultado da soma está dentro do campo de definição.

Adição de operandos com sinais opostos (8 bits)



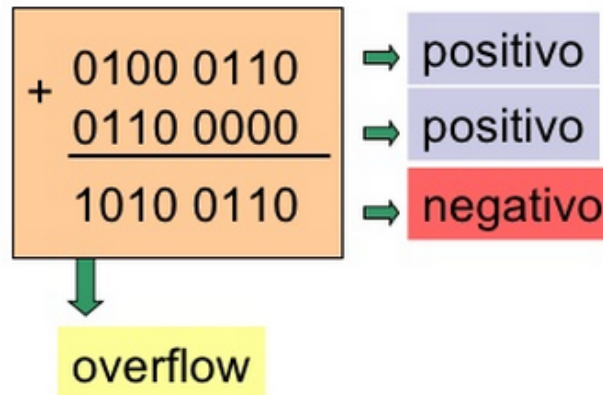
- Não ocorre overflow, o carry é ignorado e o resultado é positivo



# Detecção de Overflow

- Some os dois números e observe se ocorre o *carry* (vai 1) **sobre** o bit de sinal e se ocorre o *carry* **após** o bit de sinal.
- Se ocorrer **um e somente um** dos dois *carry*, então **houve estouro**; caso contrário o resultado da soma está dentro do campo de definição.

## Adição de 2 operandos positivos (8 bits)

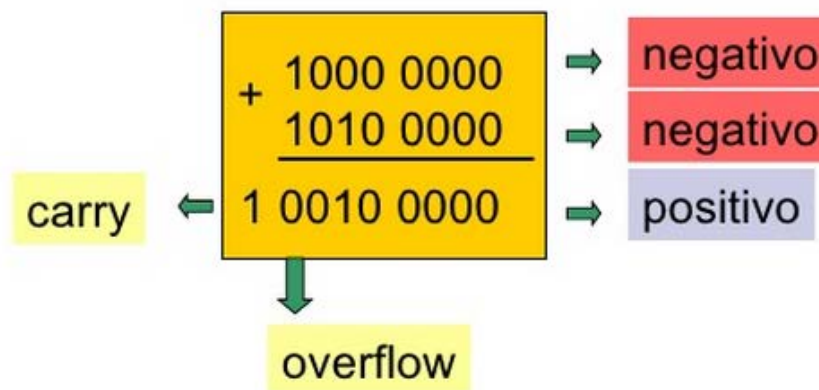


- Isto significa que o resultado está correto se o bit de sinal for ignorado

# Detecção de Overflow

- Some os dois números e observe se ocorre o *carry* (vai 1) **sobre** o bit de sinal e se ocorre o *carry* **após** o bit de sinal.
- Se ocorrer **um e somente um** dos dois *carry*, então **houve estouro**; caso contrário o resultado da soma está dentro do campo de definição.

## Adição de 2 operandos negativos (8 bits)



- Isto significa que o resultado é negativo e está em complemento a 2

# Detecção de Overflow

**EX:** Considere as operações em 4 bits

pré-definidos (MSB = bit de sinal:

*0* → positivo / *1* → negativo):

- Some os dois números e observe se ocorre o *carry* (vai 1) **sobre** o bit de sinal e se ocorre o *carry* **após** o bit de sinal.
- Se ocorrer **um e somente um** dos dois *carry*, então **houve estouro**; caso contrário o resultado da soma está dentro do campo de definição.

$$\begin{array}{r} 0011 \text{ (3)} \\ + 0010 \text{ (2)} \\ \hline 0101 \text{ (5)} \end{array}$$

$$\begin{array}{r} 0100 \text{ (4)} \\ + 0101 \text{ (5)} \\ \hline 1001 \text{ (-7)} \end{array} \quad \text{OVERFLOW}$$

→ (9)

$$\begin{array}{r} 0101 \text{ (5)} \\ + 0110 \text{ (6)} \\ \hline 1011 \text{ (-5)} \end{array} \quad \text{OVERFLOW}$$

→ (11)

$$\begin{array}{r} 1111 \text{ (-1)} \\ + 1100 \text{ (-4)} \\ \hline 1011 \text{ (-5)} \end{array}$$

$$\begin{array}{r} 1100 \text{ (-4)} \\ + 1011 \text{ (-5)} \\ \hline 0111 \text{ (7)} \end{array} \quad \text{OVERFLOW}$$

→ (-9)

$$\begin{array}{r} 1001 \text{ (-7)} \\ + 0110 \text{ (6)} \\ \hline 1111 \text{ (-1)} \end{array}$$

# Detecção de Overflow

Ex:  $5_{10} + 3_{10} = 8_{10}$  (utilização de 4 bits)

$$\begin{array}{r}
 0101_2 \\
 +0011_2 \\
 \hline
 1000_2
 \end{array}$$

Diagram illustrating the addition of 5 (0101<sub>2</sub>) and 3 (0011<sub>2</sub>) in 4-bit binary. The result is 1000<sub>2</sub>, which is circled in red. An arrow points from the circled result to the next diagram.

Com o acréscimo de um bit seria possível

$$01000_2$$

Diagram showing the result of the addition with an additional bit, 01000<sub>2</sub>, where the leading 0 is circled in red.

Utilizando-se 4 bits, o número 1000 em C2 é o  $-8_{10}$ , e não o  $8_{10}$

Notar: quando o bit mais significativo for 1, trata-se de um número negativo. No caso desse exemplo seria necessário mais um bit para representar 8 usando a representação binária em complemento de dois.

Além disso...

quando usamos complemento de dois com padrões de quatro bits (um para o sinal), ao valor 9 não está associado padrão algum; por isso não conseguimos obter uma resposta certa para a soma  $5 + 4$ , o resultado apareceria como -7.

# Mais Exemplos...

$$\begin{array}{r}
 0101 \quad 5 \\
 + 1010 \quad -6 \\
 \hline
 1111 \quad -1
 \end{array}$$

Não houve *Carry* = **não overflow**

$$\begin{array}{r}
 0110 \quad 6 \\
 + 1011 \quad -5 \\
 \hline
 0001 \quad 1
 \end{array}$$

*Carry* sobre o “bit de sinal” e após ele  
= **não overflow**

$$\begin{array}{r}
 1011 \quad -5 \\
 + 1010 \quad -6 \\
 \hline
 0101 \quad -11
 \end{array}$$

*Carry* somente após o “bit de sinal” =  
**overflow**

$$\begin{array}{r}
 0101 \quad 5 \\
 0010 \quad 2 \\
 + \quad \quad \quad \\
 \hline
 0111 \quad 7
 \end{array}$$

Não houve *Carry* = **não overflow**

# Outras Operações

# Adição em BCD

A adição de códigos BCD é semelhante a adição binária sem sinal. A correção é realizada com 6 (em binário).

$\begin{array}{r} 4 \quad 0100 \\ + 5 \quad 0101 \\ \hline 9 \quad 1001 \end{array}$	$\begin{array}{r} 5 \quad 0101 \\ + 9 \quad 1001 \\ \hline 14 \quad 1110 \\ + 0110 \text{ — correction} \\ \hline 10+4 \quad 1 \ 0100 \end{array}$
--	--

Some os seguintes números BCD:

- a)  $0011_2$  e  $0100_2$
- b)  $00100011_2$  e  $00010101_2$
- c)  $010001010111_2$  e  $010100010111_2$

# Adição em Octal

Semelhante ao sistema decimal, considerando que deve-se realizar o empréstimo na base correspondente, 8 para octal e 16 para hexadecimal.

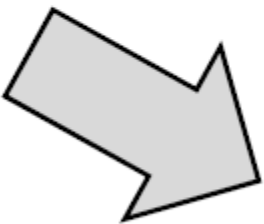
11110

5746<sub>8</sub>

+ 3472<sub>8</sub>

---

11440<sub>8</sub>



OCTAL

1	1	1	1	0
	5	7	4	6
+	3	4	7	2
	9	12	12	8
	8+1	8+4	8+4	8+0
1	1	4	4	0

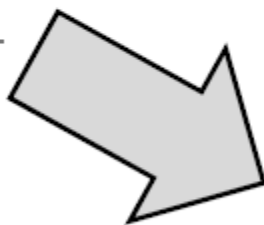
Verifique para  
número decimal !!!



# Adição em Hexa

Semelhante ao sistema decimal, considerando que deve-se realizar o empréstimo na base correspondente, 8 para octal e 16 para hexadecimal.

$$\begin{array}{r}
 1100 \\
 19B9_{16} \\
 + C7E6_{16} \\
 \hline
 E19F_{16}
 \end{array}$$



HEXADECIMAL

	1	1	0	0
	1	9	11	9
+	12	7	14	6
<hr/>				
	14	17	25	15
	14	16+1	16+9	15
	E	1	9	F

Verifique para  
número decimal !!!

# Subtração em Octal / Hexa

Semelhante ao sistema decimal, considerando que deve-se realizar o empréstimo na base correspondente, 8 para octal e 16 para hexadecimal.

$\begin{array}{r} 5746_8 \\ - 3472_8 \\ \hline 2254_8 \end{array}$	➔	<table border="0" style="margin: auto;"> <tr> <td></td><td>0</td><td>-1</td><td>8</td><td>0</td></tr> <tr> <td></td><td>5</td><td>7</td><td>4</td><td>6</td></tr> <tr> <td>-</td><td>3</td><td>4</td><td>7</td><td>2</td></tr> <tr> <td></td><td>2</td><td>2</td><td>5</td><td>4</td></tr> </table>		0	-1	8	0		5	7	4	6	-	3	4	7	2		2	2	5	4				
	0	-1	8	0																						
	5	7	4	6																						
-	3	4	7	2																						
	2	2	5	4																						
$\begin{array}{r} 9B3_{16} \\ - 7E6_{16} \\ \hline 1CD_{16} \end{array}$	➔	<table border="0" style="margin: auto;"> <tr> <td></td><td></td><td>-1</td><td></td></tr> <tr> <td></td><td>-1</td><td>16</td><td>16</td></tr> <tr> <td></td><td>9</td><td>11</td><td>3</td></tr> <tr> <td>-</td><td>7</td><td>14</td><td>6</td></tr> <tr> <td></td><td>1</td><td>12</td><td>13</td></tr> <tr> <td></td><td>1</td><td>C</td><td>D</td></tr> </table>			-1			-1	16	16		9	11	3	-	7	14	6		1	12	13		1	C	D
		-1																								
	-1	16	16																							
	9	11	3																							
-	7	14	6																							
	1	12	13																							
	1	C	D																							

# Subtração em Hexa via Binário

$$CA_{16} - 7D_{16} = ?? \text{ (8 bits)}$$

$$CA_{16} = 11001010_2 \text{ e } 7D_{16} = 01111101_2$$

Logo após, aplicamos o mesmo processo já visto:

Complemento de 2 de 01111101:

$$10000010 \Rightarrow 10000010 + 1 \Rightarrow 10000011$$

Operação:

$$\begin{array}{r} 11001010 \\ +10000011 \\ \hline 01001101 \end{array}$$

Após obtido o resultado, o transformamos em hexadecimal:

$$01001101_2 = 4D_{16}$$

$$\therefore CA_{16} - 7D_{16} = 4D_{16}$$

# Sistemas de Numeração

## Exercícios (Conversão de Base)

1 - Converta para o sistema decimal:

- |                |                        |
|----------------|------------------------|
| a) $100110_2$  | e) $11000101_2$        |
| b) $011110_2$  | f) $11010110_2$        |
| c) $111011_2$  | g) $011001100110101_2$ |
| d) $1010000_2$ |                        |

2 - Converta para o sistema binário:

- |               |                 |
|---------------|-----------------|
| a) $78_{10}$  | e) $808_{10}$   |
| b) $102_{10}$ | f) $5429_{10}$  |
| c) $215_{10}$ | g) $16383_{10}$ |
| d) $404_{10}$ |                 |

# Sistemas de Numeração

## Exercícios (Conversão de Base)

1 - Converta para o sistema decimal:

a) $100110_2$	38	e) $11000101_2$	197
b) $011110_2$	30	f) $11010110_2$	214
c) $111011_2$	59	g) $011001100110101_2$	13.109
d) $1010000_2$	80		

2 - Converta para o sistema binário:

a) $78_{10}$	100 1110	e) $808_{10}$	11 0010 1000
b) $102_{10}$	110 0110	f) $5429_{10}$	1 0101 0011 0101
c) $215_{10}$	1101 0111	g) $16383_{10}$	11 1111 1111 1111
d) $404_{10}$	1 1001 0100		

# Sistemas de Numeração

## Exercícios (Conversão de Base)

3 - Quantos bits necessitaríamos para representar cada um dos números decimais abaixo?

a)  $512_{10}$

b)  $12_{10}$

c)  $2_{10}$

d)  $17_{10}$

e)  $33_{10}$

f)  $43_{10}$

g)  $7_{10}$

4 - Converta para o sistema decimal os seguintes números hexadecimais:

a)  $479_{16}$

b)  $4AB_{16}$

c)  $BDE_{16}$

d)  $F0CA_{16}$

e)  $2D3F_{16}$

# Sistemas de Numeração

## Exercícios (Conversão de Base)

3 - Quantos bits necessitaríamos para representar cada um dos números decimais abaixo?

a) $512_{10}$	10	e) $33_{10}$	6
b) $12_{10}$	4	f) $43_{10}$	6
c) $2_{10}$	2	g) $7_{10}$	3
d) $17_{10}$	5		

4 - Converta para o sistema decimal os seguintes números hexadecimais:

a) $479_{16}$	1.145	d) $F0CA_{16}$	61.642
b) $4AB_{16}$	1.195	e) $2D3F_{16}$	11.583
c) $BDE_{16}$	3.038		

# Sistemas de Numeração

## Exercícios (Conversão de Base)

5 - Converta os seguintes números decimais em hexadecimais:

a)  $486_{10}$

b)  $2000_{10}$

c)  $4096_{10}$

d)  $5555_{10}$

e)  $35479_{10}$

6 - Converta para o sistema binário:

a)  $84_{16}$

b)  $7F_{16}$

c)  $3B8C_{16}$

d)  $47FD_{16}$

e)  $F1CD_{16}$

7 - Converta para o sistema hexadecimal os seguintes números binários:

a)  $10011_2$

b)  $1110011100_2$

c)  $100110010011_2$

d)  $11111011110010_2$

e)  $1000000000100010_2$



# Sistemas de Numeração

## Exercícios (Conversão de Base)

5 - Converta os seguintes números decimais em hexadecimais:

- |                |             |                 |             |
|----------------|-------------|-----------------|-------------|
| a) $486_{10}$  | <b>1E6</b>  | d) $5555_{10}$  | <b>15B3</b> |
| b) $2000_{10}$ | <b>7D0</b>  | e) $35479_{10}$ | <b>8A97</b> |
| c) $4096_{10}$ | <b>1000</b> |                 |             |

6 - Converta para o sistema binário:

- |                |                          |                |                            |
|----------------|--------------------------|----------------|----------------------------|
| a) $84_{16}$   | <b>1000 0100</b>         | d) $47FD_{16}$ | <b>100 0111 1111 1101</b>  |
| b) $7F_{16}$   | <b>111 1111</b>          | e) $F1CD_{16}$ | <b>1111 0001 1100 1101</b> |
| c) $3B8C_{16}$ | <b>11 1011 1000 1100</b> |                |                            |

7 - Converta para o sistema hexadecimal os seguintes números binários:

- |                     |            |                         |             |
|---------------------|------------|-------------------------|-------------|
| a) $10011_2$        | <b>13</b>  | d) $11111011110010_2$   | <b>3EF2</b> |
| b) $1110011100_2$   | <b>39C</b> | e) $1000000000100010_2$ | <b>8022</b> |
| c) $100110010011_2$ | <b>993</b> |                         |             |

# Operações Aritméticas

## Exercícios (Operações com Binários)

1 - Efetue as operações:

- a)  $1000_2 + 1001_2$
- b)  $10001_2 + 11110_2$
- c)  $101_2 + 100101_2$
- d)  $1110_2 + 1001011_2 + 11101_2$
- e)  $110101_2 + 1011001_2 + 1111110_2$

2 - Resolva as subtrações, no sistema binário:

- a)  $1100_2 - 1010_2$
- b)  $10101_2 - 1110_2$
- c)  $11110_2 - 1111_2$
- d)  $1011001_2 - 11011_2$
- e)  $100000_2 - 11100_2$

# Operações Aritméticas

## Exercícios (Operações com Binários)

1 - Efetue as operações:

a)  $1000_2 + 1001_2$  **1 0001**

b)  $10001_2 + 11110_2$  **10 1111**

c)  $101_2 + 100101_2$  **10 1010**

d)  $1110_2 + 1001011_2 + 11101_2$  **111 0110**

e)  $110101_2 + 1011001_2 + 1111110_2$  **1 0000 1100**

2 - Resolva as subtrações, no sistema binário:

a)  $1100_2 - 1010_2$  **10**

b)  $10101_2 - 1110_2$  **111**

c)  $11110_2 - 1111_2$  **1111**

d)  $1011001_2 - 11011_2$  **11 1110**

e)  $100000_2 - 11100_2$  **100**

# Operações Aritméticas

## Exercícios (Operações com Binários)

3 - Multiplique:

a)  $10101_2 \times 11_2$

d)  $11110_2 \times 110_2$

b)  $11001_2 \times 101_2$

e)  $100110_2 \times 1010_2$

c)  $110110_2 \times 111_2$

4 - Represente os números  $+97_{10}$  e  $-121_{10}$ , utilizando a notação sinal-módulo.

5 - Estando o número 10110010 em sinal-módulo, o que ele representa no sistema decimal?

6 - Determine o complemento de 1 de cada número binário:

a)  $01110100_2$

b)  $11000010_2$

# Operações Aritméticas

## Exercícios (Operações com Binários)

3 - Multiplique:

a)  $10101_2 \times 11_2$  **11 1111**

d)  $11110_2 \times 110_2$  **1011 0100**

b)  $11001_2 \times 101_2$  **111 1101**

e)  $100110_2 \times 1010_2$  **1 0111 1100**

c)  $110110_2 \times 111_2$  **1 0111 1010**

4 - Represente os números  $+97_{10}$  e  $-121_{10}$ , utilizando a notação sinal-módulo.

5 - Estando o número 10110010 em sinal-módulo, o que ele representa no sistema decimal?

6 - Determine o complemento de 1 de cada número binário:

a)  $01110100_2$

b)  $11000010_2$

# Operações Aritméticas

## Exercícios (Operações com Binários)

7 - Represente os seguintes números na notação do complemento de 2:

- |                 |                 |
|-----------------|-----------------|
| a) $1011_2$     | d) $11010100_2$ |
| b) $100001_2$   | e) $01010011_2$ |
| c) $10111101_2$ |                 |

8 - Qual o equivalente em decimal do número  $10110111_2$ , aqui representado em complemento de 2?

9 - Efetue as operações utilizando o complemento de 2:

- |                            |                               |
|----------------------------|-------------------------------|
| a) $101101_2 - 100111_2$   | d) $-10010011_2 + 11011010_2$ |
| b) $10000110_2 - 110011_2$ | e) $-10011101_2 - 1000101_2$  |
| c) $111100_2 - 11101011_2$ |                               |

10 - Efetue em binário as operações, utilizando a aritmética do complemento de 2:

- |                        |                         |
|------------------------|-------------------------|
| a) $44_{16} - 3E_{16}$ | c) $-BC_{16} + FC_{16}$ |
| b) $A9_{16} - E0_{16}$ | d) $-22_{16} - 1D_{16}$ |

# Operações Aritméticas

2.9 Add the following pairs of octal numbers:

$$\begin{array}{r} \text{(a)} \quad 1372 \\ + \quad 4631 \\ \hline \end{array}$$

$$\begin{array}{r} \text{(b)} \quad 47135 \\ + \quad 5125 \\ \hline \end{array}$$

$$\begin{array}{r} \text{(c)} \quad 175214 \\ + \quad 152405 \\ \hline \end{array}$$

$$\begin{array}{r} \text{(d)} \quad 110321 \\ + \quad 56573 \\ \hline \end{array}$$

2.10 Add the following pairs of hexadecimal numbers:

$$\begin{array}{r} \text{(a)} \quad 1372 \\ + \quad 4631 \\ \hline \end{array}$$

$$\begin{array}{r} \text{(b)} \quad 4F1A5 \\ + \quad B8D5 \\ \hline \end{array}$$

$$\begin{array}{r} \text{(c)} \quad F35B \\ + \quad 27E6 \\ \hline \end{array}$$

$$\begin{array}{r} \text{(d)} \quad 1B90F \\ + \quad C44E \\ \hline \end{array}$$

2.7 Add the following pairs of binary numbers, showing all carries:

$$\begin{array}{r} \text{(a)} \quad 110101 \\ + \quad 11001 \\ \hline \end{array}$$

$$\begin{array}{r} \text{(b)} \quad 101110 \\ + \quad 100101 \\ \hline \end{array}$$

$$\begin{array}{r} \text{(c)} \quad 11011101 \\ + \quad 1100011 \\ \hline \end{array}$$

$$\begin{array}{r} \text{(d)} \quad 1110010 \\ + \quad 1101101 \\ \hline \end{array}$$

# Dúvidas ??







OBRIGADO PELA ATENÇÃO

Prof. Victor M. Miranda